EDT

EDT Editor Manual

Order No. AA-M476A-TK

digital™ software

EDT Editor Manual

Order No. AA-M476A-TK

September 1983

This manual describes how to use the EDT interactive text editor and serves as a reference source. The manual is intended for all users of EDT.

SOFTWARE VERSION: EDT V3.00

digital equipment corporation · maynard, massachusetts

First Printing, September 1983

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

Copyright © 1983 by Digital Equipment Corporation
All Rights Reserved.

Printed in U.S.A.

A postpaid READER'S COMMENTS form is included on the last page of this document. Your comments will assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	DIBOL	RSX
DEC/CMS	EduSystem	UNIBUS
DEC/MMS	IAS	VAX
DECnet	MASSBUS	VMS
DECsystem-10	PDP	VT
DECSYSTEM-20	PDT	
DECUS	RSTS	digital
DECwriter		

ZK2479

HOW TO ORDER ADDITIONAL DOCUMENTATION

In Continental USA and Puerto Rico call 800-258-1710

In New Hampshire, Alaska, and Hawaii call 603-884-6660

In Canada call 613-234-7726 (Ottawa-Hull)
800-267-6146 (all other Canadian)

DIRECT MAIL ORDERS (USA & PUERTO RICO)*

Digital Equipment Corporation
P.O. Box CS2008
Nashua, New Hampshire 03061

*Any prepaid order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575)

DIRECT MAIL ORDERS (CANADA)

Digital Equipment of Canada Ltd.
940 Belfast Road
Ottawa, Ontario K1G 4C2
Attn: A&SG Business Manager

DIRECT MAIL ORDERS (INTERNATIONAL)

Digital Equipment Corporation
A&SG Business Manager
c/o Digital's local subsidiary or approved distributor

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Northboro, Massachusetts 01532

Contents

Preface *xiii*

Summary of New Features and Technical Changes *xiv*

Chapter 1 Introduction to EDT

1.1 Overview 1-1
1.2 An Overview of the EDT Editing Session 1-2
1.3 Types of EDT Commands 1-2
1.3.1 Keypad Editing Functions 1-2
1.3.2 Line Editing Commands 1-2
1.3.3 Nokeypad Editing Commands 1-2
1.3.4 The SET and SHOW Commands 1-3
1.4 Special Features 1-3
1.4.1 The HELP Facility 1-3
1.4.2 The Journal Facility 1-3
1.4.3 Startup Command Files 1-3
1.4.4 Key Definitions for Keypad Editing Keys 1-4
1.4.5 Macros 1-4
1.4.6 Tabbing 1-4

Chapter 2 EDT Basics

2.1 Introduction 2-1
2.2 Basic Terminology and Concepts 2-1
2.2.1 EDT Definitions 2-1
2.2.2 Editing Modes 2-4
2.2.3 Editing Session 2-4
2.2.4 Prompts 2-5
2.2.4.1 System Prompts 2-5
2.2.4.2 EDT Prompts 2-5
2.2.5 Files 2-6
2.2.6 Input and Output Files 2-6
2.2.7 File Specification 2-7
2.2.8 Buffers 2-7
2.2.9 Text Entities 2-9
2.2.10 Control Characters 2-9
2.2.11 EDT's Position and Direction 2-10
2.2.12 Strings 2-11
2.2.13 Select Ranges 2-12
2.2.14 Line Numbers 2-12
2.2.15 Line Ranges 2-13

2.3 Special Keys on Your Terminal 2-14
2.4 Starting and Ending Your EDT Session 2-14
2.4.1 How to Begin 2-15
2.4.2 Ending Your EDT Session 2-16
2.5 Recovering Your EDT Session – The Journal Facility 2-17

Chapter 3 Keypad Editing

3.1 Introduction 3-1
3.2 Definitions of Terms Used in Chapter 3 3-1
3.3 The Keyboard and Keypad 3-2
3.3.1 VT100-type Keyboard 3-2
3.3.2 VT52 Keyboard 3-3
3.3.3 LK201 Keyboard 3-3
3.3.4 Using the Keypad 3-3
3.4 Starting Your EDT Session 3-6
3.5 Ending Your EDT Session 3-8
3.6 The HELP Facility 3-10
3.7 Refreshing the Screen – CTRL/W 3-12
3.8 Entering Text in an Empty File 3-12
3.9 Moving the Cursor 3-14
3.10 Deleting Text 3-18
3.10.1 Deleting and Undeleting Characters – DEL C, DELETE, UND C 3-18
3.10.2 Deleting and Undeleting Words – DEL W, LINEFEED, UND W 3-20
3.10.3 Deleting and Undeleting Lines – DEL L, DEL EOL, CTRL/U, UND L. 3-24
3.10.4 Other Uses for the Delete and Undelete Functions 3-29
3.11 Adding Lines to Your Buffer – RETURN and OPEN LINE 3-30
3.12 Locating Strings in Your Buffer – FIND and FNDNXT 3-32
3.12.1 FIND 3-32
3.12.2 Editing the Search String or Cancelling the Search 3-34
3.12.3 FNDNXT (Find Next) 3-35
3.12.4 Examples Using FIND and FNDNXT 3-35
3.13 Functions That Manipulate Blocks of Text – SELECT, RESET, CUT, PASTE, APPEND, REPLACE 3-36
3.13.1 SELECT 3-36
3.13.2 Using RESET to Cancel a Select Range 3-37
3.13.3 CUT 3-37
3.13.4 PASTE 3-38
3.13.5 APPEND 3-42
3.13.6 REPLACE 3-45
3.14 Other Keypad Functions That Use the Select Range – CHNGCASE and FILL .. 3-46
3.14.1 CHNGCASE 3-46
3.14.2 FILL 3-48
3.15 Substituting in Keypad Mode – SUBS 3-49
3.16 Using EDT's Preset Tab Stops – TAB 3-52
3.17 Inserting Special Characters in Your Text – SPECINS 3-53

3.18 The Keypad Repeat and Backward Functions – GOLD + [–][n] 3-55
3.19 Using Commands from Other EDT Modes – COMMAND, CTRL/Z 3-57
3.19.1 Using Line Mode Commands from Keypad Mode 3-58
3.19.2 Shifting to Line Mode for a Series of Commands 3-59

Chapter 4 Line Editing

4.1 Introduction 4-1
4.2 Definitions of Terms Used in Chapter 4 4-2
4.3 Using Commands in Line Editing 4-2
4.3.1 Specifiers 4-3
4.3.2 Qualifiers 4-3
4.4 Buffer and Range Specifiers 4-4
4.4.1 The Buffer Specifier 4-4
4.4.2 The Range Specifier 4-5
4.4.2.1 Period 4-6
4.4.2.2 Number 4-6
4.4.2.3 Strings 4-6
4.4.2.4 BEGIN 4-7
4.4.2.5 END 4-7
4.4.2.6 LAST 4-8
4.4.2.7 WHOLE 4-8
4.4.2.8 BEFORE 4-8
4.4.2.9 REST 4-8
4.4.2.10 SELECT 4-9
4.4.2.11 Specifying Line Numbers That Do Not Exist 4-9
4.4.2.12 Using Range Specifiers with the <null> Command 4-9
4.4.2.13 Error Messages with Line Ranges 4-10
4.5 Editing a File with Line Mode 4-10
4.6 The Commands That End Your EDT Session – EXIT, QUIT 4-10
4.7 The HELP Facility – HELP 4-11
4.8 Inserting and Deleting Lines – INSERT, DELETE 4-13
4.9 Moving from Place to Place – FIND, <null>, TYPE 4-16
4.10 Performing Substitutions – SUBSTITUTE, SUBSTITUTE NEXT 4-18
4.10.1 SUBSTITUTE 4-19
4.10.2 SUBSTITUTE NEXT 4-21
4.10.3 Comparing SUBSTITUTE and SUBSTITUTE NEXT 4-22
4.11 Line Editing in Action – Sample Session 4-23
4.12 Using Different Buffers in Line Mode 4-28
4.12.1 SHOW BUFFER 4-29
4.12.2 FIND, <null>, and TYPE with a Buffer Specifier 4-30
4.12.3 Deleting a Buffer with CLEAR 4-30
4.13 Moving Blocks of Text – COPY, MOVE, and REPLACE 4-31
4.13.1 MOVE 4-31
4.13.2 COPY 4-33
4.13.3 REPLACE 4-35

4.14 Additional Information about DELETE, INSERT, and REPLACE 4-36
4.14.1 DELETE 4-36
4.14.2 INSERT 4-37
4.14.3 REPLACE 4-38
4.15 Commands That Use External Files – INCLUDE, PRINT, WRITE 4-38
4.15.1 INCLUDE 4-39
4.15.2 WRITE 4-40
4.15.3 PRINT 4-41
4.16 Line Numbering Your Text – RESEQUENCE 4-41
4.17 Formatting Text – FILL 4-43
4.18 TAB 4-44
4.19 Using Several Line Mode Commands on a Single Line 4-45
4.20 QUALIFIERS in Line Mode 4-46
4.20.1 /BRIEF 4-46
4.20.2 /NOTYPE 4-47
4.20.3 /QUERY 4-47
4.20.4 /STAY 4-49

Chapter 5 Nokeypad Editing

5.1 Introduction 5-1
5.2 Definitions of Terms Used in Chapter 5 5-2
5.3 Basics of Nokeypad Commands 5-2
5.4 Nokeypad Specifiers 5-3
5.4.1 The Buffer Specifier 5-3
5.4.2 The Count Specifier 5-3
5.4.3 The Sign (+ | –) Specifier 5-4
5.4.4 The String Specifier 5-4
5.4.5 The Entity Specifier 5-5
5.4.5.1 The String Entity 5-9
5.4.5.2 The V Entity 5-9
5.5 Using Nokeypad Editing 5-11
5.5.1 Starting to Edit in Nokeypad Mode 5-12
5.5.2 The Three Forms of the Line Mode CHANGE Command 5-13
5.5.3 Ending Your Nokeypad Editing Session 5-13
5.5.4 Using Line Mode Commands in Nokeypad Mode 5-15
5.5.5 Getting Online Help about Nokeypad Editing 5-16
5.6 Editing Text in Nokeypad Mode 5-17
5.6.1 Inserting Text – I 5-17
5.6.2 Moving the Cursor – The "move" Command 5-18
5.6.3 Deleting Text – D 5-20
5.6.4 The Undelete Commands – UNDC, UNDW, UNDL 5-22
5.6.5 Combining the Delete and Insert Functions – R 5-24
5.6.6 Performing Substitutions – S, SN 5-24
5.6.7 Commands That Change Case – CHGC, CHGL, CHGU, DLWC, DMOV, DUPC 5-27

5.6.8 Using the Select Range – SEL, DESEL, TGSEL, SSEL 5-29
5.6.8.1 SEL 5-29
5.6.8.2 DESEL 5-30
5.6.8.3 TGSEL 5-30
5.6.8.4 SSEL 5-30
5.6.9 Moving and Copying Text – CUT, PASTE, APPEND, KS 5-31
5.6.9.1 CUT 5-31
5.6.9.2 PASTE 5-32
5.6.9.3 APPEND 5-35
5.6.9.4 KS 5-36
5.7 Commands That Affect Your Editing Session 5-37
5.7.1 Commands That Change EDT's Direction – ADV, BACK 5-37
5.7.2 Nokeypad Commands That Affect the Screen Display – REF, SHL, SHR, TOP 5-38
5.7.2.1 REF 5-38
5.7.2.2 TOP 5-39
5.7.2.3 SHL and SHR 5-39
5.7.3 Clearing the Search String – CLSS 5-40
5.7.4 Sounding the Terminal Bell – BELL 5-40
5.8 Formatting Text with the FILL Command 5-40
5.9 Special Nokeypad Commands – ASC, ^, DATE 5-41
5.9.1 ASC 5-41
5.9.2 The Circumflex (^) 5-42
5.9.3 DATE 5-43
5.10 Joining Nokeypad Commands on a Single Command Line 5-43

Chapter 6 The SET and SHOW Commands

6.1 Introduction 6-1
6.2 Summary of SET and SHOW Commands 6-1
6.3 SET/SHOW ENTITY 6-6
6.3.1 SET ENTITY WORD 6-7
6.3.2 SET ENTITY SENTENCE 6-8
6.3.3 SET ENTITY PARAGRAPH 6-8
6.3.4 SET ENTITY PAGE 6-9
6.4 SET/SHOW SEARCH 6-9
6.4.1 Parameters That Match Case and Diacritical Marks 6-10
6.4.2 Cursor Location 6-11
6.4.3 Search Limits 6-11
6.5 SHOW BUFFER 6-12
6.6 SHOW KEY 6-14

Chapter 7 Advanced EDT Facilities

7.1 Introduction 7-1
7.2 Startup Command Files 7-1
7.2.1 Creating a Startup Command File 7-2
7.2.2 Sample Startup Command Files 7-2
7.2.3 Using SET COMMAND and SHOW COMMAND in a Startup Command File 7-10
7.3 Defining and Redefining Keys 7-11
7.3.1 Basics 7-12
7.3.2 What Keys Can Be Defined 7-12
7.3.2.1 Keypad Keys 7-13
7.3.2.2 Control Keys 7-13
7.3.2.3 GOLD Control Keys 7-14
7.3.2.4 GOLD Keyboard Keys 7-14
7.3.2.5 Special Keyboard Keys 7-14
7.3.3 Parentheses, Periods, and Nokeypad Commands in Keypad Definitions 7-15
7.3.4 Using Prompts in Key Definitions 7-17
7.3.5 The Key Definition Process 7-20
7.3.5.1 Keypad Mode: CTRL/K 7-20
7.3.5.2 Editing and Testing Your CTRL/K Definitions 7-26
7.3.5.3 Line Mode: DEFINE KEY 7-26
7.3.5.4 Special Characters in Key Definitions 7-31
7.3.6 Key Definitions at Work 7-32
7.3.7 Creating a Table with User-Defined Keys 7-37
7.4 Defining EDT Macros 7-42
7.5 Shifting Editing Modes 7-49
7.5.1 Moving from Line Mode to the Change Modes 7-49
7.5.1.1 Making a Complete Shift from Line Mode to a Change Mode 7-49
7.5.1.2 Using a Nokeypad Command Line from Line Mode 7-50
7.5.2 Shifting from Keypad Mode to Line or Nokeypad Mode 7-50
7.5.2.1 Keypad to Line Mode 7-50
7.5.2.2 Keypad to Nokeypad 7-52
7.5.3 Shifting from Nokeypad Mode to Keypad Mode or Line Mode 7-54
7.5.3.1 Nokeypad to Line Mode 7-54
7.5.3.2 Nokeypad to Keypad 7-55
7.6 The EDT Journal File 7-55
7.6.1 Looking at the Journal File 7-55
7.6.2 Editing the Journal File 7-59

7.7 EDT's Tabbing Facility 7-62
7.7.1 Tabbing Operations in the Screen Modes 7-62
7.7.1.1 Using the TAB Function to Arrange Text in Columns 7-63
7.7.1.2 Creating Layered Text 7-66
7.7.1.3 CTRL/E – The Tab Increment Function 7-66
7.7.1.4 CTRL/D – The Tab Decrement Function 7-67
7.7.1.5 CTRL/A – The Tab Compute Function 7-67
7.7.1.6 CTRL/T – The Tab Adjust Function 7-67
7.7.1.7 Using Screen Mode Tab Functions to Create an Outline 7-68
7.7.2 Tabbing in Line Mode 7-73
7.8 Using Hardcopy Change Mode 7-75
7.8.1 Editing in Hardcopy Change Mode 7-76
7.8.2 A Sample Editing Session in Hardcopy Change Mode 7-77
7.9 Using EDT Line Numbers 7-80
7.9.1 Properties of Line Numbers 7-81
7.9.2 EDT Line Numbering with Existing Files 7-81
7.9.3 VFC Record Format 7-82
7.9.4 Using the /SEQUENCE Qualifier 7-82
7.9.5 Line Numbers with the PRINT Command 7-83
7.10 Creating or Modifying the HELP File 7-84
7.10.1 HELP File Structure 7-85
7.10.2 HELP File Format 7-87
7.10.3 Formatting the Keypad Mode HELP Text 7-87
7.10.4 Using Different HELP Files in Your EDT Session 7-89

Chapter 8 EDT Command Reference

8.1 Introduction 8-1
8.2 EDT Commands, Qualifiers, and Specifiers 8-4

Appendix A EDT Messages

Appendix B EDT Command Summary

B.1 KEYPAD COMMANDS B-1
B.2 LINE COMMANDS B-5
B.3 NOKEYPAD COMMANDS B-9

Appendix C Terminal Support and Interface for EDT

C.1 Terminals That EDT Supports C-1
C.2 Terminal Interface C-1
C.3 Terminals Operating at Low Data Transmission Rates C-2

Appendix D Using EDT with the VAX/VMS Operating System

D.1 Calling Up EDT D-1
D.2 Information on Using Files with EDT D-1
D.3 EDIT /EDT Command Qualifiers D-2
D.4 Control Characters That Cannot be Defined D-6
D.5 Terminal Information and Settings Under the VAX/VMS Operating System D-6
D.6 Accessing the Same EDTINI File in All Subdirectories D-7
D.7 EDT Default File Attributes D-7
D.8 Information on EDT HELP Files D-8
D.9 Work File Location D-8
D.10 The Temporary File with EXIT, PRINT, and WRITE D-8
D.11 Callable EDT D-9
D.11.1 Calling Sequences for FILEIO, WORKIO, XLATE D-11
D.11.2 XLATE D-15
D.11.3 Callable EDT Example in VAX-11 BASIC D-16

Appendix E Using EDT with the RSTS/E Operating System

E.1 Calling Up EDT E-1
E.2 RSTS/E Files with EDT E-2
E.3 EDT Default File Attributes E-2
E.4 DCL Command Qualifiers for EDT E-3
E.5 CCL Command Qualifiers and Specifiers for EDT E-6
E.6 Control Characters That Cannot be Defined E-9
E.7 Terminal Settings with EDT Under the RSTS/E Operating System E-9
E.8 The Tentative File with EXIT, PRINT, and WRITE E-10
E.9 Information on EDT HELP Files E-10
E.10 Chainable EDT E-10
E.10.1 Requirements for the Program Chaining to EDT E-11
E.10.2 Using the /CHAIN Qualifier E-12
E.10.3 Using the /CCL Qualifier E-12
E.10.4 Failure of Monitor Directives E-13
E.10.5 Privileged Programs E-13

Appendix F Using EDT with the RSX-11M and RSX-11M-PLUS Operating Systems

F.1 Calling Up EDT F-1
F.2 DCL – EDT Command Line Qualifiers and Specifiers F-1
F.3 MCR – EDT Command Line Qualifiers and Specifiers F-4
F.4 EDT Default File Attributes F-7
F.5 Default Output Files F-8
F.6 Control Characters That Cannot be Defined F-8
F.7 Using Terminals with EDT and the RSX-11M/M-PLUS Systems F-8
F.8 The Temporary File with EXIT, PRINT, and WRITE F-9
F.9 Information on EDT HELP Files F-10
F.10 Running EDT on an RSX-11M Operating System F-10
F.11 The Location of the Work File F-10
F.12 Calling EDT From a Program on RSX-11M/M-Plus Systems F-11

Appendix G DEC Multinational Character Set

Tables:

4-1: Range Types 4-5
4-1: Range Specifier Symbols and Words 4-5
4-3: Comparison of SUBSTITUTE and SUBSTITUTE NEXT Commands 4-22
4-4: Commands that Move EDT's Position in Your Editing Session 4-30
4-5: Line Mode Qualifiers 4-46
5-1: Nokeypad Entities 5-5
6-1: SET and SHOW Command Summary 6-2
7-1: Definable Keys and Their Characteristics/Restrictions 7-13
7-2: Preset Keypad Key Definitions 7-21
7-3: EDT Preset CTRL (control) and GOLD Key Definitions 7-23
7-4: EDT Preset Keyboard Key Definitions 7-23
7-5: EDT Preset Definitions for Keys and Key Sequences on the LK201 Keyboard ... 7-24
7-6: Keypad and Nokeypad Tabbing Commands 7-62
7-7: DEFINE KEY Numbers /HELP KEY Numbers 7-88
8-1: EDT Cross Reference Chart 8-2
D-1: VAX/VMS EDT Command Qualifiers D-3
E-1: RSTS/E DCL EDIT Command Qualifiers E-3
E-2: RSTS/E CCL EDT Command Qualifiers E-8
F-1: RSX DCL EDIT/EDT DCL Command Qualifiers F-2
F-2: RSX MCR EDT Command Qualifiers F-7

Figures:

3-1: The VT100 Keyboard 3-4
3-2: The VT52 Keyboard 3-4
3-3: The LK201 Keyboard 3-4
3-4: Keypad Editing Keys – VT100 Terminals 3-5
3-5: Keypad Editing Keys – VT52 Terminals 3-5
3-6: VT100 Keypad HELP Diagram 3-11
3-7: VT52 Keypad HELP Diagram 3-11
6-1: EDT Keypad Key Numbers 6-15
6-2: FUNCTION Key Numbers for the LK201 Keyboard 6-15
7-1: Keypad Key Numbers for Use with DEFINE KEY and SHOW KEY 7-28
7-2: Additional Keypad Editing Keys – LK201 Keyboard 7-29
7-3: FUNCTION Key Numbers for the LK201 Keyboard 7-29

Preface

Audience

EDT, DIGITAL's standard interactive text editor, is designed for use on these operating systems: VAX/VMS, RSTS/E, RSX-11M, and RSX-11M-PLUS. The manual you are reading, the *EDT Editor Manual*, describes how to use EDT to create and edit text files interactively. This manual consists of explanations and examples of EDT editing operations and tasks, as well as a reference section, which discusses each EDT editing command and function.

The manual is intended for all levels of users – beginners through experienced. You must have:

- Access to an operating system that has the EDT editor.
- Knowledge of how to use the RETURN key to send information to the computer.
- Familiarity with the terminal you will be using with EDT.
- Knowledge of the basic system commands for your operating system, such as logging on and off, and using elementary system commands that list your file directory as well as those that delete, copy, or rename files.

The *EDT Editor Manual* covers all aspects of EDT. However, new users with no previous computer text editing experience will find the following manual helpful in learning how to use EDT:

Introduction to the EDT Editor

For information on your operating system, consult the documentation for that system.

Using This Manual

The *EDT Editor Manual* can be divided into three major sections:

1. User's Guide
2. Advanced EDT
3. Reference Information

The first section – Chapters 1 - 6 – contains tutorial and task-oriented information on most of EDT's commands and functions. The second section – Chapter 7 – covers the advanced EDT features in detail. The third section – Chapter 8 and the appendixes – contains reference material.

You might find it convenient to divide this book into three smaller manuals so that you are using only one section at a time. Depending on your knowledge of EDT, you will find some chapters more useful than the others. It is recommended, however, that everyone read Chapter 2 to become familiar with the basic concepts of the EDT editor.

Document Structure

Chapter 1:	Provides a broad introduction to EDT.
Chapter 2:	Provides information on basic computer text editing concepts as they apply to EDT. These concepts include information on how EDT works, how to start and end your EDT session, files and buffers, text entities, strings, ranges, and line numbers. The last section describes EDT's journal facility, which enables you to recover your work if a system interruption occurs.
Chapter 3:	Presents an in-depth description of the major keypad editing functions and examples showing how to use them.
Chapter 4:	Describes line editing commands and techniques. Some line mode commands are useful even if you plan to use keypad editing most of the time. This chapter includes a full discussion of line ranges.
Chapter 5:	Describes nokeypad editing and includes a complete table of nokeypad entity specifiers. Nokeypad mode is used primarily to create key definitions for keypad mode.
Chapter 6:	Contains a reference table for all the SET and SHOW commands as well as detailed descriptions of several of the more complex and frequently used SET and SHOW commands.
Chapter 7:	Presents information on advanced features of EDT. The topics covered include startup command files, key definitions, macro definitions, tabbing, shifting modes, EDT line numbers, and modification of the EDT HELP file.
Chapter 8:	Describes all EDT commands, qualifiers, and specifiers. The items are arranged alphabetically, each starting at the top of a page. Directly under the item name is its editing mode: keypad, line, or nokeypad. Examples are given for most items. A cross-reference chart at the beginning of the chapter groups commands in all three editing modes by their editing function.
Appendix A:	Contains all the EDT messages that can appear during an editing session. There is a brief explanation for each message.
Appendix B:	Presents a command summary, arranged by editing mode. Use this appendix to quickly refresh your memory about a command's syntax, or to find out what commands are available in a particular mode.
Appendix C:	Gives details on terminal support for EDT. Check here to find out if you can use keypad or nokeypad mode with your terminal.
Appendix D:	Includes information specific to using EDT on a VAX/VMS operating system. Qualifiers for the VAX/VMS EDIT command are described. This appendix tells you how to call EDT from a running program on a VAX/VMS system.
Appendix E:	Includes information specific to using EDT on a RSTS/E operating system. Qualifiers and specifiers for the RSTS/E EDIT and EDT commands are described. This appendix tells you how to chain to EDT from a running program on a RSTS/E system.

Appendix F: Includes information specific to using EDT on an RSX-11M or RSX-11M-PLUS operating system. Qualifiers and specifiers for the RSX EDIT and EDT commands are described. This appendix tells you how to spawn EDT from a running program.

Appendix G: Lists the DEC Multinational Character Set. Includes the decimal equivalent values for each character in the set.

Conventions

This manual uses the following conventions and symbols:

Symbol		Description
Cursor	▢	A rectangle represents the cursor in screen editing examples.
Control Character Sequence	CTRL/x	The phrase CTRL/x (control x) indicates a control key sequence. First press the key labelled CTRL; then, while holding down that key, press the keyboard character key. Examples: CTRL/Z or CTRL/].
GOLD Character Sequence	GOLD/x	The phrase GOLD/x indicates a GOLD key sequence. Press the GOLD keypad key and then a keyboard key. Examples: GOLD/T or GOLD/*. The GOLD key is the upper left-most key on the numeric keypad. It is labelled PF1 on VT100-type terminals; it is blue on VT52 terminals.
Delete Key	DELETE	The DELETE key (sometimes labelled RUBOUT or ⟨X⟩) removes the character to the left of the cursor on the screen terminals. On hardcopy terminals it deletes characters starting with the one to the left of the print head.
Return Key	RETURN	The RETURN key sends an EDT command to the computer in line and nokeypad editing. When you insert text, RETURN moves the cursor or print head to the start of a new line in all three editing modes.
Spacebar	SPACE	This symbol indicates the spacebar in examples.
Space	⟨SP⟩	This symbol indicates a space in the text.
Space	▵	This symbol indicates a space in examples.
Keyboard Key	A	This is a keyboard character key.
Keypad Editing Key	DEL L UND L	This is a keypad editing key. When the key is not preceded by GOLD, the upper editing function is the one performed, in this case DEL L.
	GOLD + DEL L UND L	This is an alternate keypad editing key. Pressing GOLD and then the editing key causes the lower editing function to be performed, in this case UND L.

Command and Qualifier Abbreviations	COPY	The underlined letters indicate the minimum abbreviation for the command, in this case CO for COPY. EDT accepts any input from the minimum abbreviation through the complete command name. For clarity, examples in this manual show the complete command name.
Optional Material	[ ]	Brackets enclose optional material in command statements. This material might be a file specification, a line number, or a word. For example, [range] shows that you can enter a range of lines at that point in the command statement. Never type brackets in the actual command statement.
Alternative Command Words	{ }	Braces enclose a list of alternatives within a command. Unless the braces are contained within brackets, you must specify one of the alternatives in your command statement. The alternatives are separated from each other by vertical bars (|). Do not type braces or bars in the actual command statement.
Alternative Command Word Separator	|	The OR symbol (vertical bar) separates specifiers for the same operation or alternatives contained within braces. For example, 'string'|"string" indicates that you you can use either single or double quotation marks to surround a string.
System Prompt	☞	The hand symbol represents your operating system's command level prompt. The actual prompt symbol appears at the left of your screen or paper. The system prompts include $, >, and Ready.
User Input	red brown	Red print in examples indicates the characters that you type or the keys that you press. When keypad mode examples for VT52 terminals differ from those for VT100-type terminals, the VT52 input appears in brown.
Repetition	. . .	The horizontal ellipsis shows indefinite repetition in a command statement. For example [subtopic ...] indicates that you can specify more than one subtopic.
Omission	. . .	The vertical ellipsis indicates that some of the lines in the example or figure are not shown.

Summary of New Features and Technical Changes

EDT V3.0 has retained most of the features present in V2.0. A number of new features have been added.

Technical Changes

EDT's original line number facility has been replaced by new line numbering features. See Chapter 7 for information on editing sequence-numbered files.

In keypad mode, the VT100 editing keys GOLD + ← and GOLD + → are no longer predefined to **SHL.** (shift left) and **SHR.** (shift right) respectively. However, you can add these definitions to your editing session if you need them.

You must supply an input file specification with the INCLUDE command.

You must supply an output file specification with the PRINT and WRITE commands. If you include a file specification with your EXIT command but do not specify a file type, EDT uses the file type of the input file.

Except on a RSTS/E system, when your output file is a disk file and there is a system interruption before EDT has had a chance to complete the processing of one of these commands – EXIT, PRINT, WRITE – the incomplete copy appears in a file with the file type **.TMP** (temporary). If a writing operation is completed without interruption, the output file is renamed to have the file type you specified. On a RSTS/E system, EDT uses tentative files instead of .TMP files.

When you use the RESEQUENCE command in line mode, EDT displays a message indicating the exact number of lines that were actually renumbered. The message no longer simply indicates the number of lines that you specified in your RESEQUENCE command.

The CLEAR command deletes both the contents of a buffer and its location. After you use CLEAR, the buffer you deleted no longer appears in the SHOW BUFFER list, except in the case of MAIN or PASTE. With the MAIN and PASTE buffers, CLEAR only deletes the contents of the buffer; the names remain in the SHOW BUFFER list. When you use the CLEAR command to delete the current buffer, EDT transfers you to the MAIN buffer. If you include the CLEAR command in a key definition that locates a buffer and then uses the INCLUDE command to add an external file to that buffer, be sure to put the CLEAR command before the FIND command (for example, DEFINE KEY GOLD F AS "EXT CLEAR BUFF1; FIND =BUFF1; INCLUDE FILE.X." If the CLEAR command follows the FIND command, EDT will automatically transfer you to the MAIN buffer and include the external file in MAIN. If you have defined a buffer to be a macro, the CLEAR command not only deletes that buffer, but also removes the macro name from the list of valid line mode commands.

When you edit a file in a directory other than your current directory, EDT now puts the journal file in the current directory, not the input or output directory. You must use the /JOURNAL qualifier to have the journal file created in a different directory.

EDT now records a CTRL/C in the journal file. You can use the journal/recovery facility to restore an editing session in which you used CTRL/C to abort an EDT command.

EDT has an upper limit of 32767 for repeat counts, entity counts, and the /DUPLICATE line mode qualifier.

When EDT starts up in change mode on a VT100-type terminal, EDT sets your screen width to the width recorded in the operating system. When you exit from EDT, EDT returns the screen to this width.

EDT V3.0 contains much more error checking than EDT V2.0. As a consequence, illegal commands now produce error messages instead of undocumented effects. The most visible result of this change is in the DEFINE KEY command. You can no longer define keys by using keypad key numbers greater than 21; you must use the syntax described in this manual.

EDT V3.0 is not supported on IAS systems.

New Features

The line numbering restrictions for EDT have been changed. In most cases, the maximum number of lines is now subject to disk space restrictions, rather than EDT limits. For EDT, the new maximum line number is 2,814,749,767. More details on EDT's new line numbering features appear in Chapter 7.

There are 12 new nokeypad commands in EDT V3.0. Many of these commands have been added so that EDT can emulate the DECmail and WPS (word processing) environments. Descriptions of these new commands appear in Chapters 5 and 8. The commands are:

BELL	(sound the terminal bell)
CHGL	(change case lower)
CHGU	(change case upper)
CLSS	(clear search string)
DATE	(insert date)
DESEL	(deactivate select range)
DLWC	(default lowercase)
DMOV	(default move)
DUPC	(default uppercase)
SSEL	(search and select)
TGSEL	(toggle select)
XLATE	(pass parameter to a running program; VMS only)

A number of new SET and SHOW commands have been added. These are summarized in Table 6-1 (Chapter 6) and each command is described in Chapter 8. The new commands are:

SET {AUTOREPEAT|NOAUTOREPEAT}
SET COMMAND
SET {FNF|NOFNF}
SET HELP
SET PARAGRAPH {WPS|NOWPS}
SET PROMPT prompt-type "string"
SET {REPEAT|NOREPEAT}
SET SEARCH WPS, CASE INSENSITIVE, DIACRITICAL INSENSITIVE
SET {SUMMARY|NOSUMMARY}
SET TERMINAL [NO]SCROLL, [NO]EIGHTBIT, [NO]EDIT
SET TEXT {END|PAGE}
SET WORD {DELIMITER|NODELIMITER}

SHOW AUTOREPEAT
SHOW COMMAND
SHOW FILES
SHOW FNF
SHOW HELP
SHOW KEYPAD
SHOW LINES
SHOW MODE
SHOW NUMBERS
SHOW PARAGRAPH
SHOW PROMPT prompt-type
SHOW QUIET
SHOW REPEAT
SHOW SUMMARY
SHOW TAB
SHOW TEXT {END|PAGE}
SHOW TRUNCATE
SHOW VERIFY
SHOW WORD
SHOW WRAP

You can redefine both the GOLD keypad key and the GOLD/GOLD key sequence with the line mode DEFINE KEY command. Use the number 20 to designate the GOLD key.

EDT can now be called on VAX/VMS systems by programs. The calling program can be written in any VAX–11 language that generates calls based on the VAX–11 calling standard. Information on callable EDT appears in Appendix D.

EDT can now be chained to on RSTS/E systems. Chaining programs can be written in any RSTS/E language. Information on chainable EDT appears in Appendix E.

See Appendix F for information on how to spawn EDT from a program under the RSX-11M/M-PLUS systems.

EDT supports the DEC Multinational Character Set. See Appendix G for information on how EDT displays graphics for characters in the decimal range 128 through 255.

EDT can be used with terminals that have LK201 keyboards. Information on using EDT with these terminals appears in Chapters 3, 7, and 8 in Appendix C. You can type and display all the characters in the DEC Multinational Character set on these terminals. (See Appendix G for a complete list of the DEC Multinational Character Set.)

Chapter 1
Introduction to EDT

1.1 Overview

EDT is an interactive text editor available on these DIGITAL operating systems: VAX/VMS, RSTS/E, RSX-11M, and RSX-11M-PLUS. You can use EDT with most kinds of text files – letters, memos, or complex computer programs. With EDT you can create files, insert text into them, and edit and manipulate that text. You can also edit and manipulate text in existing files.

The following EDT features can assist you in editing text:

- Three types of editing: keypad, line, and nokeypad. You can move from one mode to another during the same editing session.
- An online HELP facility. You can use this any time during your editing session without affecting your work.
- A journal facility. The journal file protects your editing work in case of a system interruption.
- Access to files and buffers. You can work with as many files and buffers as you need during your EDT session.
- Startup command files. These enable you to preset and customize the characteristics of your editing sessions.
- The key definition facility. You can create additional keypad editing functions to customize your editing session and extend the capabilities of keypad editing.
- EDT macros. You can use EDT macros to customize your editing session and extend the capabilities of line editing as well as keypad and nokeypad editing. Macros can be saved for use in later editing sessions.
- The tabbing facility. This feature enables you to create layered text formats.

EDT has two screen modes: keypad and nokeypad. You can use these modes on both VT100-type and VT52 terminals. Check Appendix C to find out if the screen modes are supported on your terminal. Both screen modes can perform the same functions. Keypad editing has greater flexibility than nokeypad mode. Although nokeypad mode has, by default, additional editing functions, with a good understanding of nokeypad commands, you can create new definitions for keypad editing keys that take advantage of any nokeypad editing function, as well as some line mode functions. Keypad editing is described in Chapter 3; nokeypad editing is described in Chapter 5.

If your terminal cannot accommodate screen mode, you will be using line mode, which is described in Chapter 4. Screen mode users will find some line mode commands useful, particularly those that access other files and buffers during your editing session.

1.2 An Overview of the EDT Editing Session

To use EDT, you must type a system command that calls up the EDT editor and tells it the name of the file you want to edit or create.

EDT starts, by default, in line mode. As soon as you see the line mode prompt character, the asterisk (*), you are ready to begin editing. If you have a VT100-type or VT52 terminal, you can shift to screen mode to do your editing work. To shift to keypad mode, type the line mode CHANGE command. (To shift to nokeypad mode, you first type the SET NOKEYPAD command and then the CHANGE command.)

From this point on, your work can include adding and deleting text, correcting mistakes, moving text around in a variety of ways, or locating pieces of text.

When you finish your work, you have the choice of saving the new version of your work or eliminating all record of your editing session. If you want to save your work, EDT copies the text you were editing to an external file. If not, EDT simply discards the editing work.

1.3 Types of EDT Commands

EDT commands can be divided into three main groups and one subgroup. The main groups are keypad, line, and nokeypad. The SET and SHOW commands, which comprise the subgroup, are line mode commands, but they are considered separately because they can affect how EDT operates in all three modes. Keypad commands are generally called keypad functions to emphasize that you do not type letters to tell EDT to perform an operation, but instead press the key or keys that have been assigned a particular editing function.

1.3.1 Keypad Editing Functions

In keypad editing, you press keypad or keyboard keys to tell EDT which editing function to perform. The position of the cursor on the screen tells EDT where to begin performing the operation. When you start using keypad mode, you will rely on the preset editing keys. Later on, you can create your own definitions for EDT editing keys.

1.3.2 Line Editing Commands

Line editing commands consist of English words that you type in response to the asterisk prompt (*). In addition to command words, there are qualifiers and specifiers that determine how the commands are processed and the extent of the operation. The major specifier – range – tells EDT which line(s) you want the command to affect. Line mode commands use the physical text line as the basis for operation. Text manipulations are performed on a single line or group of lines.

Line mode commands enable you to move text to and from external files, as well as from one place to another within the various locations set up in your EDT session.

1.3.3 Nokeypad Editing Commands

Nokeypad commands, like line mode commands, consist of English words and abbreviations. Some commands also take specifiers. The main specifier in this mode is "entity," which tells EDT what portion of the text to process.

Nokeypad commands perform the same operations as keypad functions, but there are more entities and commands available in nokeypad editing. Because the definitions of keypad editing keys are based on nokeypad commands, you can take advantage of the additional nokeypad entities and commands by using them in new keypad key definitions.

1.3.4 The SET and SHOW Commands

SET commands are line mode commands that set a variety of operating elements for your EDT session. These commands have no effect on the text you are editing; rather, they modify the way EDT behaves. For example, there is a SET command that changes the way EDT performs searches and another that reduces the amount of text that EDT displays on the screen.

SHOW commands are line mode commands that tell you the status of various elements of your editing session. Each SET command has a corresponding SHOW command, which tells you how that particular operating element has been set. Additional SHOW commands tell you which buffers currently exist in your session, your input and output file specifications, the definitions of keypad editing keys, and the EDT version you are currently using.

1.4 Special Features

EDT's special features include online HELP information and the recovery system. You can use the key definition and macro facilities to extend EDT's capabilities. Other facilities include startup command files and formatting using the tab commands.

1.4.1 The HELP Facility

EDT's online HELP facility allows you to get help during your editing session without discontinuing your work. In keypad mode you can get information on key functions by pressing the HELP keypad key and then pressing the editing key you want to know about. For information on line and nokeypad commands, you use the line mode HELP command. Details on using the HELP facility are given in the chapters for each mode: Chapter 3 for keypad, Chapter 4 for line, and Chapter 5 for nokeypad.

1.4.2 The Journal Facility

EDT's journal facility enables you to recover your editing session if it is interrupted due to a problem at the system level. EDT saves a file containing all your editing keystrokes whenever an unexpected interruption to your work occurs. You can have EDT process this journal file to restore your work to the state it was in before the interruption. Chapter 2 describes the journal facility and shows how to use it.

1.4.3 Startup Command Files

With startup command files, you can preset various elements of your EDT session. These special files consist of line mode commands. The commands that most frequently appear in a startup command file are SET and DEFINE KEY. EDT automatically processes all the commands in the file as soon as you type the system command to start editing your text. Details on startup command files appear in Chapter 7.

1.4.4 Key Definitions for Keypad Editing Keys

You can add to the preset keypad editing keys by creating your own definitions for keypad keys or keyboard key sequences. Key definitions are based on nokeypad commands. You can combine several commands into one key definition. Chapter 7 contains information on how to define keys as well as some applications that use defined keys.

1.4.5 Macros

You can add to the list of preset line mode commands by creating macros. EDT macros are groups of line mode commands that function as a single line mode operation. The macro name becomes an EDT line mode command for the remainder of your session. When you type the macro name, EDT performs all the operations included in that macro. You can use macros to change SET commands or process a group of DEFINE KEY commands. Macros are explained in Chapter 7.

1.4.6 Tabbing

EDT has several commands that perform tabbing operations in the screen modes and one tabbing command for line mode. These commands enable you to indent lines to format text such as outlines and indented computer programs. See Chapter 7 for a description of the tabbing facility.

Chapter 2
EDT Basics

2.1 Introduction

This chapter describes basic EDT concepts and features. The first sections cover some of the terminology and concepts that you should be familiar with when using EDT. There is a brief description of starting and ending an EDT editing session. EDT's journal/recovery facility is the subject of the last section.

Because EDT has so many features, including three primary editing modes, you do not need all the information contained in this chapter before beginning your first EDT session. Users with little or no computer text editing experience should not expect to understand every detail. However, if you are aware of the topics contained in this chapter, you can refer back to them as needed. For example, when you read the section about using the journal file to recover your editing session, you might not fully grasp everything. But after a system interruption occurs and you see the journal file in your directory, you can look at the journal facility section to find out how to restore your work.

2.2 Basic Terminology and Concepts

In learning about EDT you should be aware of certain terms that are used to describe the editor and its operations. Some of these terms, such as string, file, and buffer, have the same meanings in other computer applications. Others, such as mode and editing session, have special meanings in the context of discussing EDT.

2.2.1 EDT Definitions

EDT

An interactive text editor.

Mode

Method of editing. EDT has three primary editing modes: keypad, line, and nokeypad.

Screen Mode

Either keypad or nokeypad mode. These modes can only be used on certain types of screen terminals. (See Appendix C.)

Change Mode

An editing mode that you access through the line mode CHANGE or SET MODE CHANGE command. Change modes are keypad, nokeypad, and a secondary editing mode called hardcopy change mode.

Keypad Mode

The main screen mode. This mode uses the keys located on the small keypad at the right of the terminal keyboard to perform editing functions. You can use this mode on VT100-type and VT52 terminals.

Nokeypad Mode

A screen mode that uses command words and abbreviations to perform editing operations. These commands enable you to extend the keypad mode operations through EDT's key definition facility. You can use this mode on VT100-type and VT52 terminals.

Line Mode

An editing mode that can be used on any command terminal connected to an operating system that supports EDT. Line mode uses commands to perform editing operations.

Hardcopy Change Mode

A secondary editing mode that enables you to use most nokeypad editing commands on terminals not supported for screen editing.

Editing Session

The work you do with the EDT editor from the time you type the system EDIT/EDT command until you type the EDT command EXIT or QUIT.

Insert State

The way EDT behaves when you are inserting text in line or nokeypad mode. For example, pressing the RETURN key adds a line terminator to your text instead of sending data to the computer. You must use CTRL/Z to exit from the insert state.

Operating System

The underlying software that runs the computer to which your terminal is connected. The operating systems that support EDT are: VAX/VMS, RSTS/E, RSX-11M, and RSX-11M-PLUS.

Prompt

A signal that appears on your screen or paper, indicating that either the system or EDT is ready for you to type commands or information on your terminal.

System Command Level

When you see a system prompt (for example, $, >, or Ready), respond with a system command. If you are at the system command level, you are not using the EDT editor.

File

A collection of data that is located in a relatively permanent storage area in the computer. You use EDT to create and edit text files.

Text File

A file that is composed of ASCII-type characters such as letters, digits, and punctuation marks.

Input File

A file that contains text used by EDT at the start of or during your editing session.

Output File

A file that EDT creates and puts text into.

File Specification

A unique file identifier. File specifications must include at least the file name, but can also include node, device, and directory information as well as file type and version number on some systems.

Buffer

A temporary storage area that you use during your editing session.

Text Entity

A group of one or more contiguous characters that EDT considers as a special unit. Text entities include: word, line, sentence, and page.

Control Character

A special character signal that affects printing or device operations. To send the signal to the computer, you must press both the CTRL key and a keyboard key. Control characters signal such display operations as end of line, end of page, and tab stop.

Control Key Sequence

The combination of keys needed to produce a control character. Control key sequences are represented as CTRL/x where CTRL refers to the CTRL key and **x** is a character on the keyboard.

Line Terminator

The signal that tells the computer to start the following text on a new line. In EDT you can reference the line terminator with a single character – CTRL/M – and insert a line terminator in your text by pressing the RETURN key.

Cursor

An indicator that shows EDT's current position in your text when you are using a screen mode.

Current Line

The line where EDT is currently positioned in line mode. In screen mode editing, the current line is the one on which the cursor is located.

Direction

EDT's current direction. EDT works in two directions: (1) forward toward the end or bottom of the buffer and (2) backward toward the beginning or top of the buffer.

String

A group of contiguous characters that you supply with certain EDT commands. You use strings in commands that involve searches and substitutions.

Select Range

A group of contiguous characters (often extending for several lines) that you want EDT to operate on. For example, you can delete, move, or copy a select range.

Line Number

A number that EDT automatically assigns to a line of text during your editing session. These line numbers only appear when you are working in line mode.

Line Range

A reference to one or more lines that tells EDT which part of the text you want the line mode command to affect.

Specifier

Information you supply with EDT commands to tell EDT such things as how many times to perform a command and which parts of the text to affect. Text entities, counts, and line ranges are specifiers.

Qualifier

A special line mode command word that modifies the way EDT performs the command. Qualifiers are optional. Whenever you type a qualifier, you must precede the qualifier word with a slash (for example, /QUERY).

Default

The option or setting that EDT uses when you do not explicitly indicate otherwise.

2.2.2 Editing Modes

The editing mode you use with EDT depends on the type of terminal you have and the nature of your editing work. If you have a VT100-type or VT52 terminal, you can use screen mode. (See Appendix C for a list of terminals that EDT supports in screen mode.)

You will want to focus your attention on keypad editing if you have the appropriate terminal. As you become proficient in keypad mode, you can increase the capabilities of keypad editing by creating additional keypad key definitions. Since these definitions are based on nokeypad command syntax, becoming familiar with nokeypad editing helps you extend your keypad editing functions. Chapter 3 describes keypad editing; Chapter 5 discusses nokeypad editing.

If you do not have access to a terminal that can use the screen modes, you must use line mode. Chapter 4 describes line mode editing commands. Line mode users can go directly to Chapter 4 after reading this chapter.

2.2.3 Editing Session

The term editing session or EDT session refers to the work you do at your terminal from the time you issue the system EDIT/EDT command to start editing with EDT until you give an EDT command to leave the editor. Most sessions involve inserting, locating, moving, or deleting text. For example, an EDT session might consist of asking EDT to create a new file, typing text into it, correcting mistakes or making any changes you want, and saving the results.

When you edit an existing file using EDT, you work on a copy of the original file. The original file remains in its directory and is unchanged by the editing work. At the start of the editing session, EDT places the copy in a buffer where you can edit the text. After you finish your editing session, you can save a copy of the edited text in an output file or you can save nothing. The contents of your original file are unaffected in either case.

Whenever you ask EDT to edit a file that does not exist, the editor sets up an empty buffer to hold the text you will be typing. You still have the option of saving nothing when you finish your editing session, since EDT does not actually create the new output file until you end your session.

2.2.4 Prompts

In using EDT, you will see two types of prompts: (1) a system prompt and (2) EDT prompts.

2.2.4.1 System Prompts — The system prompt is displayed by your operating system. Possible system prompts include:

```
$
>
Ready
```

Whenever you see the prompt for your system, you are ready to type a system command. The system commands associated with EDT are:

```
EDIT /EDT

EDT
```

The ☞ symbol is used in this manual to represent the system command prompt. When ☞ appears in an example, you should see either $, >, or Ready at the left of your screen or paper:

```
☞ EDIT /EDT

☞ EDT
```

On most operating systems, EDT is the standard editor. Therefore if your system uses the EDIT command, all you need to type is **EDIT**. However, if EDT is not the standard editor on your system, you must include the system command qualifier **/EDT** to be sure of getting EDT instead of some other editor.

All operating systems have a number of qualifiers or specifiers that you can use with the EDIT command. These system qualifiers enable you to do such things as change the name of the output file, use the journal/recovery facility, or examine a file that you do not have the authority to modify. In the appendix that discusses your operating system, you will find a list of the qualifiers and specifiers that you can use with the system EDIT/EDT command.

2.2.4.2 EDT Prompts — EDT has several prompts. The line mode prompt is the asterisk (*). Whenever you see the asterisk at the left of your screen or paper, you are ready to type a line mode command. The /QUERY qualifier in line mode uses the question mark (?) as a prompt. Type one of the four valid /QUERY responses when you see this prompt.

In keypad mode, EDT prompts you for a search string whenever you use the FIND function. The prompt is the phrase **Search for:** . When you use the keypad mode COMMAND function, EDT prints the prompt **Command:** to tell you to type a line mode command.

2.2.5 Files

A file is a collection of data that is treated as a distinct physical unit. Files can contain a variety of data types, but EDT can work only with text files. The type of material that is stored in text files includes letters, memos, computer programs, lists, and tables. The files are stored in the computer so that you can use them over and over again.

Text files are composed of elements such as words and numbers. The text elements are groups of characters that include letters, digits, punctuation marks, and certain signal codes (for example, line terminator, horizontal tab, and page separator). Appendix G lists all the characters that are valid in text files.

Files that contain other computer codes or signals are not considered to be text files and cannot be edited with EDT. Such files include those with special attributes. See the appendix that applies to your system to find out which file attributes are needed by EDT to edit a file.

Some nonstandard text files (for example, those that have been processed by the RUNOFF formatter) can be edited with EDT. When EDT is able to edit a nonstandard text file, this message appears as soon as you call up EDT to edit the file:

```
Input file does not have standard text format
```

You should be careful when editing these files; you might not get the results you expect. If possible, convert the files to standard text format before editing them. If you cannot convert such a file, you should be aware of which elements make the file format non-standard. Try to avoid altering those elements.

You can edit any text file that you have access to. If the file is in your current directory, simply use the file specification with the system EDIT/EDT command. For files in other directories, be sure to include the directory specification in the EDIT/EDT command line.

When you give EDT the name of a file that does not exist in your current directory or in the directory you specify, EDT prints the message:

```
Input file does not exist
```

If you are using EDT to create a new file, this message confirms that there is no file with that name. You can now enter the text you want that file to contain. When you type the EXIT command at the end of your session, EDT creates a file containing the text you typed.

If you expected EDT to find an existing file but received the **Input file does not exist** message, you have probably typed the wrong file specification or forgotten to identify the directory. Use the EDT QUIT command to stop the editing session and start over again with the system EDIT/EDT command.

2.2.6 Input and Output Files

EDT uses several files during an editing session. The term external file refers to files outside of your editing session. These include several types of input and output files.

Input files are files that already contain text that EDT uses to start your editing session or that EDT adds to your editing session. When you edit an existing file, that file is the primary input file. EDT copies the text from that file into its MAIN buffer so you can edit it.

A startup command file is another type of input file. EDT reads the information contained in that file and processes it before turning over the editing session to you.

During your editing session you can use the line mode INCLUDE command to copy other files into your editing session. These files are also input files.

Output files are those created by EDT at the end of or during your editing session. The primary output file is the one that EDT creates when you use the EXIT command to end your session. This output file contains the copy of the text in your MAIN buffer.

Using the line mode WRITE command, or PRINT command, you can create additional output files during your editing session. These output files can contain either the entire contents or a portion of any buffer. When you type the file specification with the WRITE or PRINT command, EDT creates that file in the current directory. You must include the directory specification in the command line to have EDT put the file in another directory. Note that EDT cannot create new directories. Thus if you include a directory specification as part of the file specification, that directory must exist and you must have access to it.

EDT's journal file is also an output file. EDT creates a journal file as soon as you start your editing session and continues to add information to it until the end of your session. However, in most cases, when you end your session, EDT discards the journal file. (See the last section in this chapter for more information on the journal file.)

The EDIT and EDT system commands have various qualifiers that you can use to determine whether EDT creates or uses input and output files. Information on these qualifiers appears in the system appendixes (Appendix D for VAX/VMS, Appendix E for RSTS/E, and Appendix F for RSX-11M/M+).

2.2.7 File Specification

You use the file specification to identify a specific file. A full file specification can consist of six elements:

node name
device name
directory name
file name
file type
version number

In some operating systems, file specifications do not have version numbers. In those operating systems that do, you can omit the version number whenever you want to edit the most recent version of the file or create a new file. Similarly, file specifications in some operating systems do not have node names. If your operating system does use node names, you can omit the node name whenever you are editing a file on your local node.

Whenever you are required to use a file specification in an EDT or system command, you must include at least the file name. The remaining elements are optional. Most of the examples in this manual show both the file name and the file type whenever a file specification is called for.

2.2.8 Buffers

An EDT buffer resembles a file in that it occupies storage space in the computer system and is used to contain text. However, the use of that space is temporary. EDT buffers exist only for the duration of your editing session. When you end your session, EDT discards all the buffers that were created during that session.

The first thing EDT does when you start to edit an existing file is to copy the text from that file into a buffer called MAIN. The original file still exists in its directory; EDT works only on the copy. When you use EDT to create a new file, the first thing the editor does is set aside an empty buffer called MAIN. No new file actually exists until you end your EDT session.

When you use the EXIT command to finish your editing session, EDT copies the contents of the MAIN buffer into a new file. All the buffers created during your session are deleted and the storage space they occupied is turned over to the operating system for other uses. Before you give the EXIT command, you can transfer the contents of any other buffer to an external file by using the line mode WRITE command. When you do not want to save a copy of the MAIN buffer, end your session with the QUIT command.

In addition to establishing the MAIN buffer at the start of your session, EDT also sets up another buffer called PASTE. EDT uses the PASTE buffer for screen mode CUT and PASTE functions. Both the MAIN and PASTE buffers are always part of your EDT session. You can delete the contents of these buffers during your editing session, but you cannot remove the buffers themselves until you end your session.

You can create additional buffers to store pieces of text or additional material that you might want to look at or edit during your editing session. Various line and nokeypad commands enable you to create new buffers. Some line mode commands transfer EDT to the buffer that is named in the command line. Whenever you see the buffer specifier included in the command syntax for a line mode or nokeypad mode command, you can use that command to create a new buffer.

Buffer names can contain any alphanumeric characters (letters and digits, but not punctuation marks or control characters). However, the name must begin with a letter. You can use one punctuation character, the underscore (_), in buffer names. If you like, you can create a buffer name that has over 80 characters in it. But since you will have to type the buffer name at least once, you will find short names easier to use.

The SHOW BUFFER line mode command tells you the names of the buffers that exist for your editing session and indicates the current buffer with an equal sign (=). The line mode CLEAR command can delete any buffer that you no longer need except MAIN or PASTE. In the case of MAIN or PASTE, the CLEAR command removes the contents of the buffer, but does not discard the location. After you use the CLEAR command to delete any other buffer, that buffer name no longers appears in the SHOW BUFFER list.

If you specify a buffer name in a line mode command, EDT usually transfers you to the new buffer. Should you find yourself in an empty buffer when you expected to see some text, you might wonder what happened to the text. The SHOW BUFFER command gives the number of lines in each buffer when it displays the list of buffer names. You can use this information to locate the text that you need.

EDT has some special storage areas whose names never appear in the SHOW BUFFER list because EDT maintains complete control over them. You cannot specify these buffers in commands or enter them in any way. However, you can influence the characters that are stored in them and decide when or whether to insert their contents into your text. These special buffers include the delete character buffer, the delete word buffer, the delete line buffer, the search buffer, and the substitute buffer. The three delete buffers are used by the keypad mode delete and undelete functions and the nokeypad mode delete and undelete commands. Line mode makes no use of these delete buffers.

2.2.9 Text Entities

Text entities refer to specific elements of text that EDT recognizes as units when you are working in a screen mode. These include characters, words, lines, sentences, paragraphs, as well as the beginning or end of these elements. Text entities are used in the screen modes to move the cursor and to limit the extent of operations such as deleting text. (A complete discussion of the EDT text entities appears at the beginning of Chapter 5.)

You use entities with operations that delete text, create select ranges, change the case of letters, or reformat text by means of the FILL function so that EDT knows which part of the text to work on.

You should be familiar with these entities when you start editing in a screen mode.

ENTITY	DESCRIPTION
character	Any single character: a letter, digit, punctuation mark, or control character, such as CTRL/L (form feed).
word	A group of contiguous characters that are bounded by spaces, ends of lines, tabs, or ends of pages.
line	A group of contiguous characters that begins just after a line terminator and ends with the next line terminator.
sentence	A group of contiguous characters that is bounded by a period (.), question mark (?), or exclamation point (!) provided that the punctuation mark has at least one space or a line terminator after it.
paragraph	A group of contiguous characters that is bounded on each end by two line terminators in succession.
page	A group of contiguous characters that is bounded on each end by a form feed character (CTRL/L).

In addition to the boundaries listed above, the end of your buffer (shown by the end of buffer mark – [EOB]) and beginning of your buffer are boundaries for text entities. Thus, if you have no page boundaries in your file and ask EDT to delete a page, everything in the buffer is deleted. Similarly, if you are using a text formatter like RUNOFF that often uses no blank lines in the text, when you try to delete a paragraph, EDT deletes the entire buffer.

The SET ENTITY command lets you change the boundaries of four entities: WORD, SENTENCE, PARAGRAPH, and PAGE. For instance, if there are no form feeds in your file, you can still use the page entity. You might type XXXXX every place you want a page break and then use that symbol as the page boundary in your SET ENTITY PAGE command. If you are using the RUNOFF formatter, you could specify **.B** as your paragraph boundary. But remember that no matter which boundaries you specify in the SET ENTITY command, the ends of your buffer are always boundaries.

2.2.10 Control Characters

Control characters are code elements that initiate, modify, or stop control operations. Some control characters affect printing operations such as vertical and horizontal tabulation, line termination, and page separation.

Generally, you use the CTRL key in combination with a keyboard key to access the control function. This combination is often referred to as a control key sequence. To use that sequence, you must first press the control key and then, while holding down the control key, press the character key.

There are several ways to include control characters in your text: pressing special terminal keys such as TAB, typing the control key's decimal number with the keypad SPECINS function or the nokeypad ASC command, or using the circumflex operation in nokeypad mode.

The primary control characters are the first 32 characters in the character set shown in Appendix G. The appendix lists the decimal value associated with each character. Decimals values for the primary control characters range from 0 – 31.

EDT uses some control keys to perform special editing functions. For example, when you are in keypad mode, pressing CTRL/Z shifts EDT to line mode. In both line mode and nokeypad mode, you press CTRL/Z to tell EDT that you are finished inserting text. The control keys that perform EDT functions are described in the chapters on the different editing modes. Entries also appear in Chapter 8 for each control key that has a preset EDT function.

2.2.11 EDT's Position and Direction

The screen modes are character editing modes. EDT focuses its attention on individual characters or groups of characters. EDT's position in your text is marked by the cursor. The cursor is a flashing indicator (either an underscore or shaded rectangle) that marks the character in your text where EDT is positioned. That character can be referred to as the cursor character or the current character.

In line mode, EDT focuses on the line as its unit of text. When you ask EDT to move to a certain place in your text, it moves to a specific line, not to a word or character or string. The only exception to this rule occurs when EDT searches for a string. Under this circumstance, EDT is positioned after that string so that a second search for that same string will not relocate the same occurrence of the string.

EDT has two directions – forward and backward. In the screen modes, forward is to the right of the cursor and backward is to the left. In all three modes, forward is toward the bottom or end of the buffer and backward is toward the top or beginning of the buffer.

EDT's normal direction is forward. In the screen modes it is possible to change the direction to backward for as long as you need to. You can then reset the direction to forward any time.

Regardless of which direction EDT is moving, it always stops when it reaches the top or bottom of the buffer. If you are editing in a screen mode and try to move the cursor down once it is already at the bottom of the buffer, EDT prints this message:

```
Advance past bottom of buffer
```

To continue editing, you must move EDT toward the beginning of the buffer.

When you are searching for a string and the cursor is in the middle of a buffer, EDT looks for the string only in the part of the buffer between the cursor and the bottom if the direction is forward or the top if the direction is backward. EDT cannot find a string that is located in the other portion of the buffer unless you specifically tell it to look in the other direction. For screen modes, you can use an editing key or command to tell EDT in which direction to search. For line mode, you can put a minus sign (-) in front of the string if you want EDT to search backward for it.

2.2.12 Strings

Strings can be one or more characters (including spaces), words, pieces of words, or even lines. The important element that makes the group a string is that the characters are contiguous, that is, one character next to another. Some examples of strings are:

```
DIGITAL                                                    infor
ing the mar                                                .However,
!                                                          ations
$495
and whenever the occasion arises, we can move on
```

Strings are sometimes used as command specifiers. For example, when you use the DEFINE KEY command, you replace the string specifier with the definition you are creating. However, you generally use strings to locate text and perform substitutions.

When you ask EDT to locate a string in one of the screen modes, it moves the cursor to that string. In line mode, EDT finds the line that contains the string and positions itself just after the string so that you can search for the next occurrence of that same string.

EDT is completely literal about strings. When you want to locate a string, EDT always moves to the next occurrence of the string it can find, not necessarily to the next occurrence you have in mind. If you make a mistake in typing your string, EDT either finds exactly what you typed or else prints the message **String was not found.**

When you type the string you want EDT to locate, try to make the reference unique so that EDT will find the exact place you want. Remember, however, that you can always ask EDT to find the string again, so you need not type long strings just to ensure that they are unique. All editing modes have ways to ask EDT to find the string a second time without having to type more than a few keystrokes.

Suppose, for example, that you want to locate the word **read** on the 25th line of your text. When you ask EDT to find the string **read**, it stops on line 14 at the word **ready**. When you give the command again, it stops on line 22 at the word **thread**. If you do not want to worry about what other examples of **read** EDT might find, you might include spaces in your string so that only the word **read** would be found: Ⓢ **read** Ⓢ . But note that if the word is at the beginning or end of line 25, it would not be surrounded by spaces, so EDT would pass right by line 25 and go on until it found the word **read** in the middle of a line.

When strings are used in line and nokeypad mode commands, they are set apart from the command words by delimiters. String delimiters are merely separating characters that surround the string, for example, /string/. The delimiting character cannot be a letter, a digit, or any character that appears in the string itself. You cannot use either percent sign (%) or underscore (_) as the delimiter in the line mode SUBSTITUTE or SUBSTITUTE NEXT command. You must use the same delimiter character to mark each end of the string.

In some instances you must use quotation marks as the delimiter. You can use either single quotes (') or double quotes ("), but you must use the same mark on both sides of the string. Also, if a single quotation mark appears within the string itself, you must use the double quotation marks as the delimiter, and vice versa. For example:

```
"don't"              not              'don't'
'letter "A"'         not              "letter "A""
```

In substitution commands where two strings are entered side by side, there are three delimiters separating the two strings (for example, /1982/1983/). In these cases, all three delimiters must be the same and neither string can contain the delimiter character. The usual delimiter character for substitutions is the slash (/), but any convenient nonalphanumeric character will work (except % and _ in line mode).

Here are some examples of both search and substitute strings marked with delimiters:

```
"plati"                                    >30978>4758>
|health of |                               'July 27, 1983'
/gold/GOLD/                                !chairman!chairperson!
(half-(semi-(                              -and/or-or-
```

EDT has several ways to search for strings. Left to its default settings, EDT performs searches in this manner:

1. It pays no attention to whether letters in the string are typed in upper- or lowercase (thus it considers **WRITE**, **Write**, and **write** to be identical).
2. EDT ignores diacritical marks when searching (for example, the string **Lowe** is considered the same as the string **Löwe**).
3. In the screen modes, EDT moves the cursor to the first character in the search string.
4. EDT always begins its search at the current cursor position or current line and then moves toward either the bottom of the buffer or the top, depending on the current direction or the direction specified in the search command.

You can use the SET SEARCH command to modify the way EDT performs searches. Chapter 6 contains a detailed explanation of the SET SEARCH command.

2.2.13 Select Ranges

Select ranges are used in the screen modes to tell EDT the portion of text to operate on. Select ranges can be as small as a single character, although that would be rare, or as large as thousands of lines. They can start or end anywhere on a line. Like strings, however, select ranges must be composed of contiguous characters.

In general you use select ranges when you want EDT to copy, move, or delete large chunks of text. You first set up the select range; then you tell EDT what to do with it.

To set up a select range, you must mark one end of that range with the SELECT editing key in keypad mode, or the SEL command in nokeypad mode. Then move the cursor to the other end of the text you want selected. Now you are ready to have EDT perform an operation on the marked text. Using the line mode SELECT range specifier, you can use select ranges with line mode commands such as COPY, MOVE, DELETE, and WRITE.

2.2.14 Line Numbers

EDT automatically assigns a number to every line in each buffer in your editing session. Even if you are working in one of the screen modes, EDT continues to maintain its system for numbering each line in the text.

If you are editing an existing file, EDT assigns line numbers to the text when it copies the text into the MAIN buffer. These numbers start with 1 and increase by 1. Unless you specify otherwise, EDT does not retain the line numbers when you use the EXIT command to end your editing session. Each time you edit that file, the lines are numbered starting with 1 and increasing (or incrementing) by 1.

Whenever you insert text into a buffer or use the line mode INCLUDE command to add text, EDT assigns numbers to these new lines. Thus, every line in every buffer in your EDT session has a number. New lines of inserted text often have numbers with decimal fractions, for example, 16.1 for a line inserted between lines 16 and 17.

You can renumber the text lines in a buffer during your EDT session with the line mode RESEQUENCE command. Resequencing can make it easier to keep track of the EDT line numbers. Also, when you renumber the lines, you can eliminate decimal fractions and thus make it simpler to use the line numbers as line mode range specifiers.

If you are editing an existing file that has sequence numbers, EDT converts those sequence numbers into the EDT line numbers for that session. Sequence numbers are 16-bit binary numbers attached to each record of a sequenced file. Like EDT line numbers, they are not part of the text. If there are any instances where the sequence numbers are not in ascending order, EDT adjusts the line numbers to ensure that all line numbers in a buffer are unique and ascending.

If you use the line mode INCLUDE command to insert a sequence numbered file into your text, EDT disregards the file's sequence numbers.

You can use the /SEQUENCE qualifier with the line mode EXIT and WRITE commands to have EDT convert its line numbers to sequence numbers when the editor copies text to an output file. The sequence numbers do not appear when you merely display the file at the terminal.

There are various upper limits to the number of lines EDT can handle. The maximum number of lines you can have in a buffer is 2,814,749,767 (2**48/10**5-1). The maximum sequence number that EDT can read in or write out is 65535 (2**16-1). More information about editing files with sequence numbers appears in Chapter 7.

2.2.15 Line Ranges

Most line mode commands use the range specifier to tell EDT which portion of the text to act on. You use line ranges in commands that copy, delete, insert, locate, move, replace, resequence, substitute, type, or write out text.

In some commands, the line range is only a single line. For example, the INCLUDE command needs only one line number to tell EDT where to insert the new text. In other instances, you might want to supply several blocks of text in a single command line by using two or more range specifiers in a row (DELETE 4 THRU 6, 15 THRU 20, 42, 55 THRU END).

There are a variety of ways to express line ranges besides using the line numbers themselves. For example, you can refer to a line by typing a string that appears on that line. Other range specifiers enable you to perform line mode operations from the screen modes without having to know the exact line numbers. Chapter 4 gives details on all the ways to reference line ranges.

There are some limits to the number of lines that can be specified with certain line mode commands. These limits are included in the command descriptions in Chapter 8.

2.3 Special Keys on Your Terminal

EDT uses some of the special keys on your terminal. All three editing modes use the RETURN key and DELETE key, located at the right of the main keyboard, as well as the control key (CTRL), generally located near the left SHIFT key. Both screen modes use the arrow keys to move the cursor around on the screen.

You use the RETURN key to send commands to EDT. Each time you finish typing a line or nokeypad mode command, you must press RETURN to send the information to EDT for processing.

When you are inserting text, you press the RETURN key to have EDT insert a line terminator in your text. A line terminator tells the computer to print the characters following the line terminator on the next line of your screen or paper. The line terminator is always at the end of a line. When you move the cursor in screen mode you will notice it stopping on the "invisible" line terminator at each end of line.

In line mode, the DELETE key deletes one character at a time on the line you are currently typing. It cannot delete text on lines that have already been typed. The DELETE key deletes the character to the left of either the print head on a hardcopy terminal or the cursor on a screen terminal.

In the screen modes, DELETE always removes the character to the left of the cursor. If the cursor is at the beginning of a line of text, pressing the DELETE key eliminates the preceding line terminator. When you are typing commands or search strings, you can use the DELETE key to remove characters on the command or search line. (LK201 keyboards use the symbol ⟨X⟩ for the DELETE key.)

The arrow keys are used in both screen modes to move the cursor from one place in the text to another. The arrow keys always move the cursor in the direction of the arrow, regardless of whether EDT's current direction is forward or backward.

As you might expect, keypad mode uses those keys that form the numeric keypad, the rectangular group of keys located apart from and to the right of the main keyboard. In addition to the 18 keypad keys on VT100-type terminals, the four arrow keys located at the top right of the main keyboard are considered part of the keypad, making a total of 22 keypad keys. (For VT100-type terminals that have LK201 keyboards, the arrow keys are located at the bottom of the second keypad.) There are 19 keypad keys on VT52s. (See the keypad diagrams in Chapter 3.)

Three other keys, BACKSPACE, LINEFEED, and TAB, have special editing functions in keypad mode. The functions are explained in Chapter 3. (For VT100-type terminals that have LK201 keyboards, the F12 key on the function key row corresponds to BACKSPACE. The F13 key corresponds to LINEFEED.) In line and nokeypad modes, these keys can be used insert their respective control characters into your text (CTRL/H for BACKSPACE, CTRL/I for TAB, and CTRL/J for LINEFEED).

2.4 Starting and Ending Your EDT Session

This section contains a brief explanation of how to "get in and out of" EDT. The method you use to call on the EDT editor to edit a file depends on which operating system you are using.

VAX/VMS	RSX-11M
RSTS/E	RSX-11M-PLUS

Details on each operating system's EDIT/EDT command appear in the appropriate system appendixes.

Appendix D	VAX/VMS
Appendix E	RSTS/E
Appendix F	RSX-11M and RSX-11M-PLUS

The EDT command you use to end your session – either EXIT or QUIT – depends on whether or not you want to save a copy of the MAIN buffer text. Chapter 4 contains complete information on the line mode EXIT and QUIT commands.

2.4.1 How to Begin

There are three versions of the system command that calls up the EDT editor.

```
☞ EDIT /EDT

☞ EDIT

☞ EDT
```

Generally you will want to type the file specification of the file you are editing on the same line as the system command. If you do not, you are prompted for the file specification. The examples below show several ways to call up EDT to edit the file LIST.DAT. Some include the prompt for a file specification. Only a few of these apply to your operating system. (Remember that your input is printed in red. All commands are shown with uppercase letters, but you can type them in lower or mixed case.)

```
☞ EDIT /EDT LIST.DAT

☞ EDIT LIST.DAT

☞ EDT LIST.DAT

☞ EDIT /EDT
_File:  LIST.DAT

☞ EDIT /EDT
File:           LIST.DAT

☞ EDIT
File:           LIST.DAT

☞ EDT
EDT>LIST.DAT
```

After you type the appropriate system command and file specification, EDT prints the first line of your file, followed by the line mode asterisk prompt (*) at the beginning of the next line:

```
☞ EDIT /EDT LIST.DAT
    1       List of People Attending the New Year's Party
*
```

If you are using EDT to create a new file, EDT starts by printing the message **Input file does not exist** followed by the end of buffer symbol ([EOB]). Since there is no text in your buffer, the [EOB] symbol is displayed in place of the first line. The asterisk prompt (*) follows:

```
☞ EDIT/EDT LIST.DAT
Input file does not exist
[EOB]
*
```

Now you are ready to begin editing. If you are creating a new file, that is, if there was no original file for EDT to copy, the first thing you will do is insert some text. When you are editing an existing file, you can start with any editing operation you want.

2.4.2 Ending Your EDT Session

When you finish your editing session, you can either save or discard your editing work. To save a copy of the MAIN buffer text, use the line mode EXIT command. This command instructs EDT to copy the MAIN buffer text to an external file. When you use the QUIT command, EDT discards the contents of all its buffers without copying any text.

There are several reasons why you might not want or need to save your editing work. For instance, you might use EDT only to locate things in a file so you can refresh your memory, verify something, or read a piece of text. Or you might call up EDT to copy part of a file to another file. Occasionally, you might do some editing work that you do not want to keep. By using the QUIT command, you can end your editing session without creating a new file to clutter up your directory.

When you do want to save the new version of the text, you can have EDT put the MAIN buffer text in a file with a different specification from the original file you were editing. The new file specification can be given on the EXIT command line. By including directory information in the file specification, you can have EDT put the text in a different directory.

After you have typed the EXIT command, EDT displays the complete file specification for the output file, as well as the number of lines the file contains. The following examples show sample EXIT commands with file specifications that are typical for EDT's different operating systems. In each case the input file was LIST.DAT. Notice that when you use QUIT to end your session, EDT displays nothing.

```
*EXIT
XXX0:[SMITH]LIST.DAT;3 64 lines

*EXIT
XX0:[310,6]LIST.DAT;3 64 lines

*EXIT
LIST.DAT 64 lines
```

```
*EXIT PARTYLIST.DAT
XXX0:[SMITH]PARTYLIST.DAT;3  64 lines

*EXIT [JONES]DEC20.DAT
XXX0:[JONES]DEC20.DAT;3  64 lines

*QUIT
```

2.5 Recovering Your EDT Session – The Journal Facility

EDT's journal facility acts as a safety net for your editing session. It can save you hours of work should your editing session stop unexpectedly due to a system interruption. You do not need to remember all the material covered in this section in order to start using EDT. But if you experience a system interruption, you will know to look in this section to find out how to recover your editing work.

A system interruption refers to any break in the communication between your terminal and its computer. Such events include a loss of electrical power, a break in the telephone communication link (if your terminal uses a modem or telephone to access the computer), or a problem in the computer. On some operating systems it is possible for you to cause a system interruption by pressing a control key sequence.

In most cases, when a system interruption occurs, EDT saves a copy of its journal file. You can then use the journal file to restore all or most of your editing work.

While you are editing or inserting text, EDT is keeping track of every keystroke you enter at your terminal. EDT records this information in the journal file. Unless you specify otherwise, this journal file is deleted as soon as EDT processes an EXIT or QUIT command. However, when you experience a system interruption, the journal file is not deleted.

The journal file does not contain a version of your text. Rather, it contains a record of what went on during your session. Every keystroke is in there. By combining the journal file with the text that you had at the beginning of your session, you can recover your editing session to the point just before the interruption. If you are creating a new file when the interruption occurs, all EDT needs to recover your work is the journal file and any files that you inserted with the line mode INCLUDE command.

Using the journal, EDT can restore over 95% of a long editing session in a matter of minutes. (Sometimes the last few keystrokes are not yet recorded in the journal file when the system interruption occurs.)

When you do not specify otherwise, journal files have **.JOU** as the file type. The file name is the same as the file you were editing. For example, LETTER7.RNO has LETTER7.JOU as its journal file; INVENLIST.DAT has INVENLIST.JOU, and so on.

The recovery system is easy to use. First, make sure all the input files that you used during the session are available. Then add the /RECOVER qualifier to the EDIT/EDT command that you issue to start your EDT session. If you do not supply the file specification for the file you are editing after /RECOVER, the system prompts you for it. Enter the file specification of the file you were editing when the system interruption occurred. Use one of these examples, depending on your operating system.

```
☞ EDIT /EDT /RECOVER
_File: file-specification

☞ EDIT /EDT /RECOVER file-specification

☞ EDT /RECOVER
EDT>file-specification

☞ EDT /RECOVER file-specification

☞ EDIT /RECOVER
File:  file-specification

☞ EDIT /RECOVER file-specification
```

Remember, where you see the term **file-specification**, type the name of the file you were editing, not the name of the journal file. For example:

```
EDIT /EDT /RECOVER LIST.DAT
```

Now watch EDT "replay" your editing session at the terminal. If the last few edits are missing, remember that this is normal. Generally, EDT records several keystrokes at a time in the journal file. No work from earlier in your session will be missing.

When EDT finishes recovering the session, you can pick up where it leaves off and continue your work. After EDT processes the commands in the journal file, it continues to add the new keystrokes to it. If another system interruption occurs before you can issue a successful EXIT command, the instructions that were already processed from the journal file are still in the journal file. When you use the /RECOVER qualifier the second time, EDT processes the commands from both sessions because the journal file now contains both sets of commands.

EDT generally saves the journal file in the current directory. If you are editing a file from another directory, the journal file will not be in the same directory as the input file.

Sometimes EDT saves a journal file in a directory, but the file does not get used. If you find a file with the **.JOU** file type in your directory, chances are the file was saved during a brief editing session and you never used it because you had not done enough editing work to bother recovering it. These journal files generally have no use, so you can delete them from your directory.

For more advanced information on EDT's journal facility, see Chapter 7. That section explains some of the features of the journal files and shows how to remove unwanted commands from the end of the file. You can edit the journal file to recover text that you accidentally deleted from a buffer.

The /JOURNAL qualifier or journal-file specifier can be used with the system EDIT/EDT commands to specify different names for journal files. See the appropriate appendix for your operating system for information on this qualifier or specifier.

Chapter 3
Keypad Editing

3.1 Introduction

This chapter shows how to use the preset functions of keypad editing. The first section has definitions of terms applicable to keypad mode. Next comes information on the VT100 and VT52 keypads and how to access the various keypad functions. The section on using keypad editing starts with the basic set of functions needed to enter text and do straightforward editing operations. The next sections present some of the more advanced editing capabilities. The last section shows how to use the other editing modes directly from keypad mode. This feature enables you to perform EDT functions not available in keypad mode without having to leave keypad mode.

Keypad editing is available on VT100-type and VT52 terminals. It is the editing mode that you are most likely to use if you have one of these terminals. If you are not sure whether your terminal can use keypad editing, check the list in Appendix C.

Keypad editing relies on pressing keys to perform editing functions, rather than typing commands as is done in line and nokeypad modes. When you press a keypad editing key, an instruction is sent directly to the computer for processing. The action of pressing the RETURN key to process a command is incorporated in the editing key.

When you edit in keypad mode, text appears on the screen. You use the various editing keys to perform different editing operations. The preset functions can handle most editing tasks. However, once you have become familiar with the preset functions, you can use the key definition facility to extend and customize EDT to suit your particular needs. Information on how to define keys appears in Chapter 7.

3.2 Definitions of Terms Used in Chapter 3

These terms are used in describing keypad editing. Definitions of other EDT terms appear in Chapter 2.

arrow keys

The four keys that have arrows pointing in different directions. On most VT100-type terminals, these keys are located on the top right-hand side of the main keyboard. On VT52 terminals, they are located in the right-hand column of the keypad. On VT100-type terminals that have LK201 keyboards, the arrow keys are located at the bottom of the second keypad. You use the arrow keys to move the cursor.

editing key

A keypad or keyboard key (or key sequence) that sends instructions to EDT for processing.

function

A task performed by EDT when you press the appropriate key or keys or issue a command.

GOLD key

The key located at the upper left-hand corner of the keypad. On VT100-type terminals this key is labeled PF1. On VT52 terminals, it is blue.

keyboard

The front section of the terminal, containing keys that are used to type and enter information into the computer. The keyboard is divided into two blocks of keys: the main keyboard and the keypad. LK201 keyboards are divided into three blocks of keys: the main keyboard, the numeric keypad (which corresponds to the keypad used for EDT's keypad mode), and a second keypad (which contains the arrow keys as well as some other keys).

keypad

The keys located in a separate block at the right of the keyboard. On VT100-type as well as VT52 terminals, the arrow keys are considered to be part of the keypad.

key sequence

A combination of two or three keystrokes used to perform one keypad editing function.

main keyboard

Those keys that form the larger block of keys located to the left of the keypad on the terminal keyboard.

3.3 The Keyboard and Keypad

EDT supports three different keyboards for keypad mode:

VT100
VT52
LK201

Each keyboard has an EDT editing keypad and some additional keys that EDT uses to perform editing functions. EDT also uses the control key with some main keyboard keys to perform other editing functions.

3.3.1 VT100-type Keyboard

The keypad for most VT100-type terminals (Figure 3-1) contains 18 keys and is located to the right of the main keyboard. The four arrow keys located on the upper right of the main keyboard are also considered to be part of the keypad. The control key, labeled CTRL, is located next to the CAPS LOCK key on the lower left side of the main keyboard. Four other keys on the main part of the keyboard have EDT editing functions: BACKSPACE, DELETE, LINEFEED, and TAB. The TAB key is located on the left side of the main keyboard. The other three keys are located on the right side of the main keyboard.

3.3.2 VT52 Keyboard

The VT52 keypad has 19 keys, all located on the keypad to the right of the main keyboard (Figure 3-2). The control key, labeled CTRL, is located next to the CAPS LOCK key on the lower left side of the main keyboard. Four keys on the main part of the keyboard have EDT editing functions: BACKSPACE, DELETE, LINEFEED, and TAB. The TAB key is located on the upper left side of the main keyboard. The other three keys are located on the upper right side of the main keyboard.

3.3.3 LK201 Keyboard

Some terminals that have a VT100 emulation mode use the LK201 keyboard. The LK201 keyboard has a main keyboard part and two keypads. The numeric keypad, on the right edge of the keyboard, is the EDT editing keypad and corresponds to the VT100 keypad. The arrow keys are located at the bottom of the "editing" keypad, which is located between the main keyboard and the numeric keypad. On the LK201 keyboard, the TAB key is located on the upper left side of the main keyboard and is labeled **Tab**. The key with the symbol ⟨X⟩ corresponds to the DELETE key on VT100 keyboards. The BACKSPACE and LINEFEED functions are located in the function key row across the top of the keyboard. BACKSPACE is F12; LINEFEED is F13. (If your keyboard does not have the function key row labeled, count over 12 keys from the left to find the BACKSPACE key. The next key to the right has the LINEFEED function.) The control key, labeled **Ctrl**, is located next to the Lock key on the lower left side of the main keyboard. Throughout this manual it is assumed that the LK201 function keys on your terminal have been enabled. On most terminals that use the LK201 keyboard, the function keys are enabled by default. See the documentation on your terminal for more information about enabling function keys.

In discussing keypad editing, this chapter uses the VT100/VT52 terminology for keys. If you are using a terminal with an LK201 keyboard, remember that DELETE corresponds to ⟨X⟩, F12 corresponds to BACKSPACE, and F13 corresponds to LINEFEED. Otherwise, the VT100 examples and terminology apply to your keyboard. More information on using some of the additional LK201 function keys appears in Chapters 7 and 8.

3.3.4 Using the Keypad

When you enter keypad mode, the keypad keys immediately assume their editing functions. You can no longer use them to type numbers or punctuation marks. Whenever you leave keypad mode to go to line or nokeypad mode, the keypad keys revert to their normal character-producing state. However, do not use the keypad editing keys when you are typing a line mode or nokeypad mode command directly from keypad mode.

Most keypad keys have two editing functions. To use the primary function, simply press the key. The second (or alternate) function requires that you first press the GOLD key and then the key itself. The GOLD and HELP keys have only one preset function. On VT100 terminals, the arrow keys have only one preset function. (You can define the combination of GOLD and arrow key to perform an editing function.)

Figures 3-4 and 3-5 show the functions assigned to each keypad key for the VT100 and VT52 terminals. The keypad key name is printed in the upper left-hand corner of the block. Where keys have two functions, the upper (primary) function is the one that EDT performs when you press that key alone. To use the lower (alternate) function, press the GOLD key and then the other keypad key. (Do not hold the GOLD key down while pressing the editing key.) The alternate function is shown below the primary function on all keypad diagrams and examples.

Figure 3-1: The VT100 Keyboard

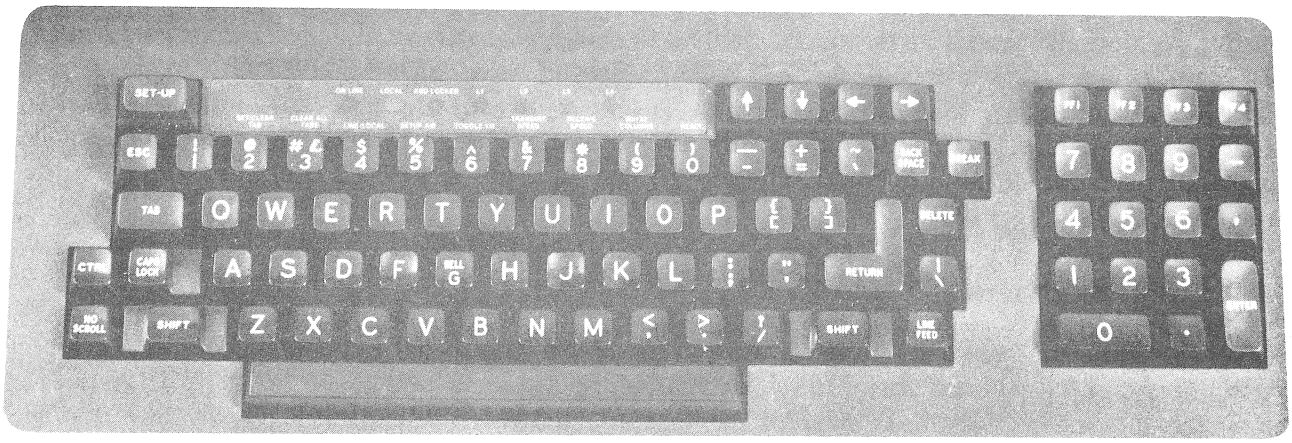


Figure 3-2: The VT52 Keyboard

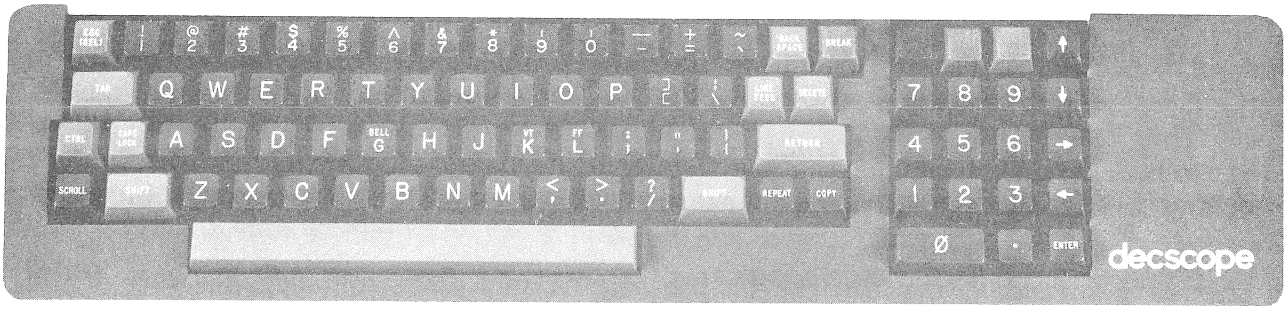


Figure 3-3: The LK201 Keyboard

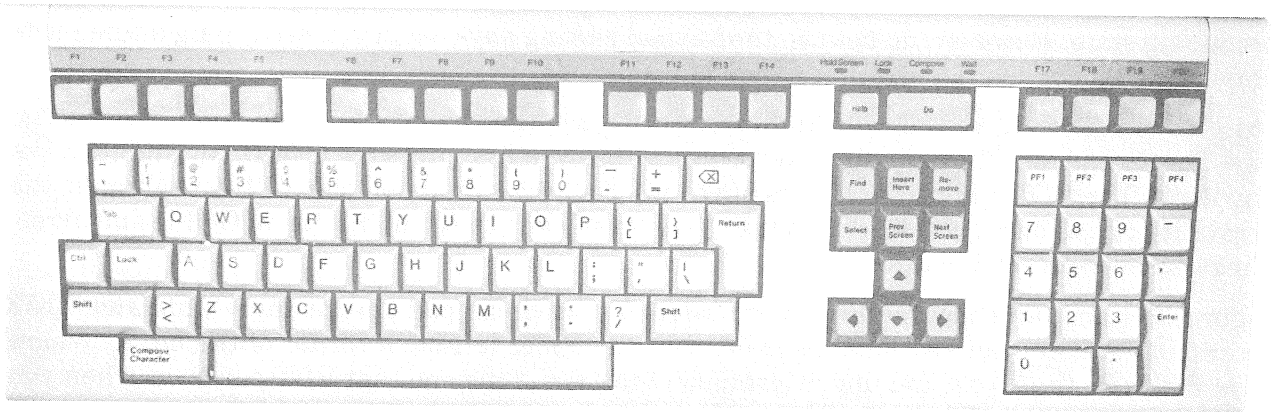
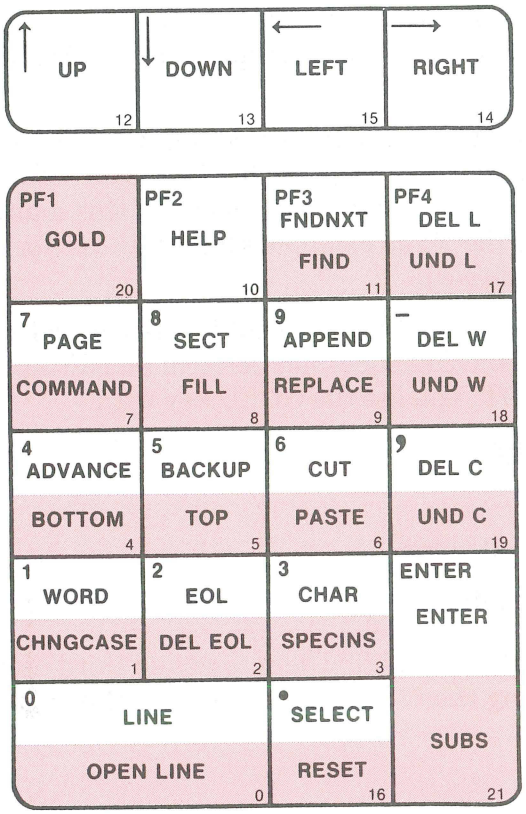
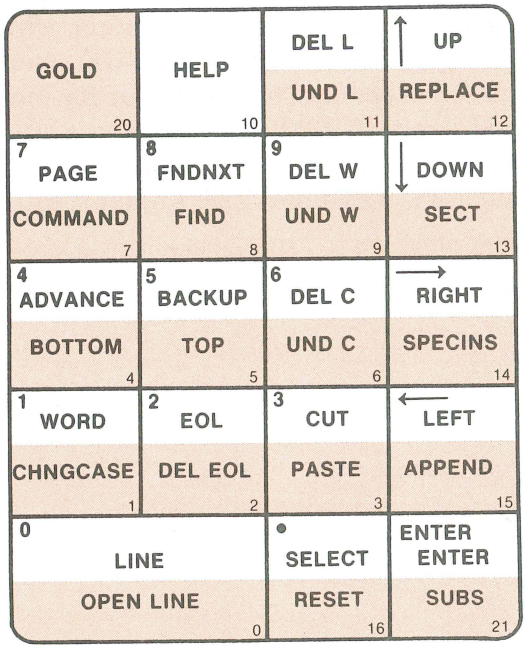


Figure 3-4: Keypad Editing Keys – VT100 Terminals



VT100

Figure 3-5: Keypad Editing Keys – VT52 Terminals



VT52

The HELP key accesses the keypad portion of the EDT HELP facility. When you press HELP (PF2 on VT100 keypads; the red key next to GOLD on VT52 keypads), EDT prints a diagram of the keypad, showing the various preset functions. It also prints a list of other keyboard and control keys that have preset keypad editing functions. A full description of the HELP facility appears in the next section.

Many editing keys are the same for VT100 and VT52 keypads, but not all. For example, to use the FIND function, you must press GOLD + PF3 on VT100s, but GOLD + 8 on VT52s. Whenever the key or key sequence is unambiguous, only one example is shown in this manual; your input is printed in red. If it is necessary to show the VT52 example separately (as, for example, with SECT), VT52 keys and input are printed in brown.

The GOLD key (located in the upper left-hand corner of the keypad) is labeled PF1 on VT100 terminals and is colored blue on VT52 terminals. GOLD does not perform an editing function, but is used to assist in sending instructions to EDT. You use the GOLD key with:

- Keypad keys to activate their alternate functions
- Digits to signal EDT to repeat a function
- Digits to enter a character using its decimal value
- Keyboard keys to form additional key definitions

The first use is of primary concern when you start using EDT. The second and third uses are discussed later in this chapter. Information on using the GOLD key with key definitions appears in Chapter 7.

3.4 Starting Your EDT Session

When you call up EDT, the default mode is line mode. If you are editing an existing file, EDT prints out the first line in that file and then gives the line mode asterisk prompt (*). EDT is now ready to accept line mode commands. However, you want to work in keypad mode. To shift to keypad mode, you must enter the line mode command CHANGE. This command, which can be abbreviated to just the letter C, shifts EDT into keypad mode. The CHANGE command has some other properties and capabilities, which are described in Chapter 8, but for most of your editing work, all you need to type is the command itself. (If EDT has access to a startup command file that includes the SET MODE CHANGE command, your editing session automatically starts in keypad mode. However, this manual always assumes EDT's default settings.)

As soon as you type CHANGE and press RETURN, EDT clears the screen completely, and displays the first 22 lines of the file you want to edit.

```
☞ EDIT /EDT OLDFILE.DAT
    1       January 10, 1984
*CHANGE
```

```
January 10, 1984


Mr. John H. Hartley
General Manager
Production Systems Department
Skyway Elevators, Inc.
28457 Main Street
Hawleyville, CT  06440


Dear Mr. Hartley:

Our group is pleased to accept your invitation to attend
a seminar on your new elevator walkways.  There was so
much interest among our people in your presentation that
I am afraid I will be bringing 23 people with me.  I hope
you can accommodate that number.

I would appreciate it if you could send us any literature
that would help us understand all the new features of
this system.
```

You are now ready to start editing the letter to Mr. Hartley. Remember that EDT has made a copy of your original file. The original still exists in your directory. When you use the line mode EXIT command to leave EDT, a copy of the edited text is put into a new file in your directory.

When you use EDT to create a new file, you enter keypad mode with the same CHANGE command. In this case, there is no text for EDT to print. All you see is the [EOB] (end of buffer) mark at the top of the screen and a message at the bottom of the screen.

```
☞ EDIT /EDT NEWFILE.DAT
Input file does not exist
[EOB]
*CHANGE
```

```
[EOB]
```

```
Input file does not exist
```

No file named NEWFILE.DAT exists in your directory at the moment. However, when you use the EXIT command to leave EDT, a copy of your text will be put into a file with that name and the new file will be in your directory.

The two previous examples show an entire screen display. A screen contains 22 lines for text. The 23rd and 24th lines are used for displaying EDT messages as well as different kinds of information you enter for EDT. The remainder of the manual shows only partial screens so that you can focus attention on the text that is affected by EDT.

3.5 Ending your EDT Session

When you finish your editing session, you must use the line mode EXIT or QUIT command to leave EDT. If you use EXIT, EDT copies the contents of the MAIN buffer to an output file. The QUIT command simply ends your editing session without saving a copy of any editing work. No new file is created. For more information on these two line mode commands, see Chapters 2 and 4.

There are two ways to access the line mode commands that you need to end your session. The first way transfers you from keypad mode back to line mode, where you can type the EXIT or QUIT command as soon as the asterisk prompt (*) appears on your screen. To shift from keypad mode back to line mode, press CTRL/Z.

```
Sincerely yours,


David R. Jones
Vice President
HiRise Construction Division

cc:  R. P. Doncaster
     T. R. Sullivan

DRJ/mq
```

CTRL/Z

```
[EOB]
*EXIT
```

The other way to end your EDT editing session is with the keypad mode COMMAND function. COMMAND is the alternate function on the 7 keypad key. To use this, first press GOLD; then 7. EDT prints the prompt **Command:** at the bottom of the screen (on the 23rd line) indicating that the editor is ready to accept a line mode command while still in keypad editing. Type the EXIT or QUIT command and press the ENTER key to send it to EDT. (If you press RETURN, EDT adds a CTRL/M character, displayed as ^M, at the end of the line and waits for more input. Simply press the DELETE key once to remove the ^M and then press ENTER to send the command to the computer for processing.) This example uses the EXIT command to end your editing session.

GOLD + PAGE COMMAND

```
cc:  R. P. Doncaster
     T. R. Sullivan

DRJ/mq


Command: EXIT

DISK$USER:[QUEENSLEY]OLDFILE.DAT;2 58 lines
☞
```

If you list your directory, you find a file named OLDFILE.DAT;1 and one named OLDFILE.DAT;2. (If you use EDT on a RSTS/E system, you find the files OLDFILE.DAT and OLDFILE.BAK.) The new version of the text you were editing has been stored in a separate output file in your directory. The original file you started with is also in your directory; it has not been changed at all by EDT.

When you edit a file from another directory, the EXIT command places the edited version of your file in the same directory as the original file. Suppose your current directory is JONES and the file you want to edit is in the SMITH directory.

```
☞ EDIT /EDT [SMITH]LETTER.DAT
     1                June 30, 1984
    .
    .
    .
*EXIT
DISK$USER:[SMITH]LETTER.DAT;2 35 lines
☞
```

No copy of the file LETTER.DAT appears in your current directory.

3.6 The HELP Facility

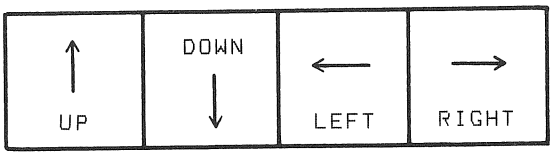
When you are editing in keypad mode, you can use the HELP key to get online information about keypad editing functions. The HELP facility contains a diagram of the keypad for your terminal as well as a list of keyboard and control keys that have preset EDT functions.

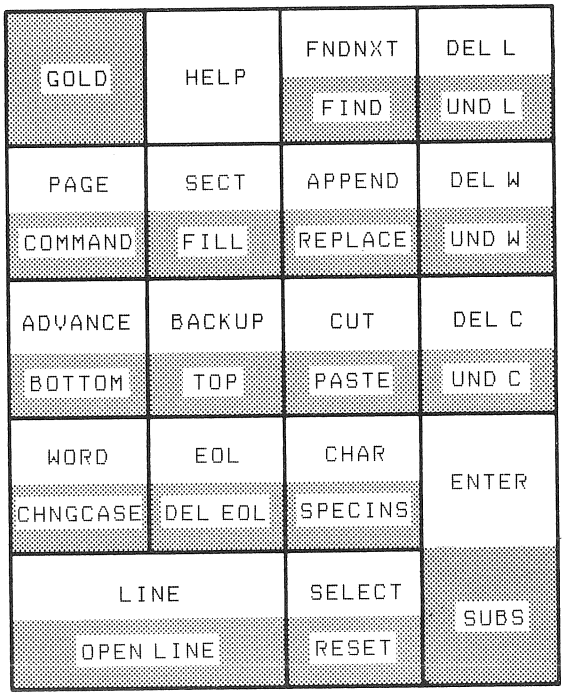
Once you have pressed the HELP key, you are in the keypad mode HELP facility. To get information on a particular keypad key, simply press that key. If you want information on a keyboard key such as DELETE or TAB, press that key. For help on a control key sequence such as CTRL/A, press the CTRL key and then press the keyboard key, for example, A. To get help on a GOLD keypad sequence, simply press the keypad key. Information for both the primary and alternate functions for that key will be displayed. For information on a GOLD keyboard key, such as GOLD W, press only the keyboard key, for example, W; do not press the GOLD key. Remember to press the HELP key before you press any other key or key sequence to get HELP information.

When you are in the HELP facility you can continue to press different keys that you want to learn about without having to return to the keypad diagram. To see the diagram again, press the RETURN key; to return to your editing work, press the spacebar.

Figures 3-6 and 3-7 show the HELP file display for VT100-type and VT52 terminals. To see the diagram for your terminal press the HELP key (PF2 on VT100; the red key on VT52).

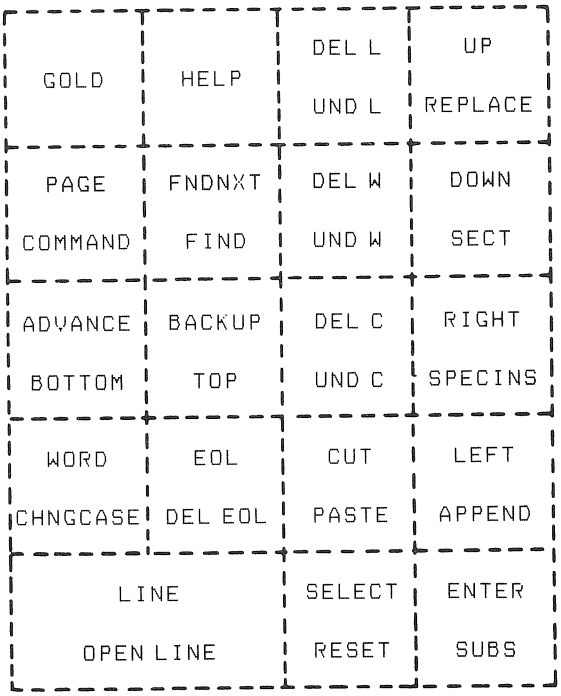
Figure 3-6: VT100 Keypad HELP Diagram






```
DELETE          Delete character
LINEFEED        Delete to beginning of word
BACKSPACE       Backup to beginning of line

CTRL/A          Compute tab level
CTRL/D          Decrease tab level
CTRL/E          Increase tab level
CTRL/K          Define Key
CTRL/R          Refresh screen
CTRL/T          Adjust tabs
CTRL/U          Delete to beginning of line
CTRL/W          Refresh screen
CTRL/Z          Return to line mode

Press a Key for help on that Key.
To exit, press the space bar.
```

Figure 3-7: VT52 Keypad HELP Diagram




```
DELETE          Delete character
LINEFEED        Delete to beginning of word
BACKSPACE       Backup to beginning of line

CTRL/A          Compute tab level
CTRL/D          Decrease tab level
CTRL/E          Increase tab level
CTRL/F          Fill text
CTRL/K          Define Key
CTRL/R          Refresh screen
CTRL/T          Adjust tabs
CTRL/U          Delete to beginning of line
CTRL/W          Refresh screen
CRTL/Z          Return to line mode

*** For help on a Key, press the Key.
*** To exit, press the spacebar.
```

3.7 Refreshing the Screen — CTRL/W

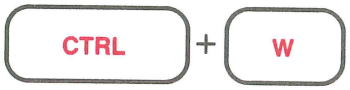
Any time during your editing session, a message from outside EDT might appear on your screen. Such a message might be sent by the system operator to tell you something important about the current or future state of the computer – for example: **System unavailable on New Year's Eve.** If you have electronic mail at your site, the message might indicate that you have mail.

These messages can be flashed on your screen right in the middle of your EDT session. In some cases the message appears to replace text on your screen. In other cases, it might duplicate the line you were working on and put the message in the middle of that text. Whatever the appearance, the message has absolutely no effect on your text.

You could ignore the extraneous characters, but often they confuse your work. EDT's CTRL/W function refreshes the screen to eliminate all the characters that are not part of your text.

This example shows only a portion of the screen. The interrupting message appears at the beginning of the third line. After you press CTRL/W, the screen reflects only the current portion of the text buffer that you were editing.

```
however, we cannot see our way clear to meeting your
deadline by the 30th of August. It looks as if we
New mail from JONES  ng ready by late September. This
estimate would also include completion of the second
```




```
however, we cannot see our way clear to meeting your
deadline by the 30th of August. It looks as if we
could have everything ready by late September. This
estimate would also include completion of the second
```

In keypad mode, CTRL/R performs the same function as CTRL/W.

3.8 Entering Text in an Empty File

To create a new file using EDT, do the following:

- Call up EDT.
- Enter keypad mode.
- Type some text.
- Edit the text.
- End your EDT session.

In keypad mode all you need to do to insert text is type the text. There is no insert command or insert editing key. Wherever the cursor is located before you begin to type is where the inserted text will start. When you want to begin a new line, simply press RETURN to move the cursor to the beginning of the next line. Then continue typing your text.

First, call up EDT to create the file NEWFILE.DAT. Remember that EDT does not actually create the new file until you type the EXIT command.

```
☞ EDIT /EDT NEWFILE.DAT
Input file does not exist
[EOB]
*CHANGE
```

After you type the CHANGE command, the only item on your screen is the end of buffer marker and the EDT message.

```
[EOB]
  .
  .
  .
Input file does not exist
```

To enter text, simply start typing. For example, type **Greetings**.

```
Greetings
[EOB]
```

Greetings is now part of your text. Notice that the cursor is positioned just after the **s**. If you want to type more text, the new material starts where the cursor is positioned.

```
Greetings!   This is your friendly EDT writer.
```

3.9 Moving the Cursor

In editing or adding to your text, you will need to move the cursor to different parts of the buffer. Keypad functions that move the cursor are:

↑	up arrow	↓	down arrow
←	left arrow	→	right arrow
FIND		**FNDNXT**	find next
PAGE		**SECT**	section
BOTTOM		**TOP**	
CHAR	character (VT100s only)	**WORD**	
LINE		**EOL**	end of line
BACKSPACE	beginning of line (F12 – LK201)		

EDT has two directions for editing: forward and backward. Some of the keys that move the cursor depend on EDT's current direction to determine whether they move the cursor to the right toward the end of the buffer or to the left toward the beginning of the buffer. Other keys always move the cursor either forward or backward regardless of the current direction. Because FIND and FNDNXT are a special case in this regard, they are discussed later in the chapter.

The keypad editing keys that rely on EDT's current direction are:

CHAR	**WORD**
EOL	**LINE**
SECT	**PAGE**

You change EDT's direction by the ADVANCE and BACKUP functions. Pressing ADVANCE (4) sets the direction to forward; pressing BACKUP (5) sets the direction to backward. You can change directions any time during your editing session.

The CHAR function (3) is available only on VT100 keypads. (If you have a VT52 terminal, you must rely on the left and right arrow keys for this function.) CHAR moves the cursor one character to the right if the current direction is forward, or to the left if the current direction is backward. If the cursor is at the end of a line on the line terminator, CHAR in the forward state moves the cursor to the beginning of the next line. Conversely, if the direction is backward and the cursor is at the beginning of the line, CHAR moves the cursor to the line terminator character at the end of the previous line.

The WORD function (1) moves the cursor to the beginning of the next word if the direction is forward regardless of whether the cursor is at the beginning, middle, or end of the previous word. If the direction is backward, the cursor is moved to the beginning of the word it is on when it is located in the middle or end of a word. Only when the cursor is at the beginning of a word does it move to the beginning of the previous word. (For more information on EDT's definition of a word, see the section on SET ENTITY WORD in Chapter 6.)

The EOL (end of line) function (2) moves the cursor to the end of the current line if the direction is forward. The cursor is positioned on the line terminator following the last character (including any spaces) in the line. If the direction is backward, the cursor is moved to the line terminator of the previous line in the text. If the cursor is already at the end of a line when you press EOL, it moves to the end of the next line or previous line depending on EDT's direction.

The LINE function (0) differs from the EOL function in that it leaves the cursor on the first character of a line, not on the line terminator. When the direction is forward, the cursor is positioned on the first character of the next line of text. If the direction is backward, the cursor rests at the beginning of the current line. When the cursor is already at the beginning of a line and the direction is backward, LINE moves the cursor to the beginning of the previous line.

The SECT (section) function (8 for VT100; GOLD + ↓ for VT52) moves the cursor 16 lines in the current direction. The cursor always rests at the beginning of a line regardless of its position before SECT was pressed.

The PAGE function (7) relies on your having page markers in the text buffer you are working on. EDT uses the form feed character (CTRL/L, displayed by EDT as ⟨FF⟩), to signal the beginning of a new page. If you have no page markers in your text, PAGE moves the cursor to either the bottom or the top of the buffer, depending on EDT's direction. (More information on EDT's definition of a page appears under the SET ENTITY PAGE command in Chapter 6.)

The functions that move the cursor in the same way regardless of EDT's current direction are:

Left Arrow (←)	**Right Arrow (→)**
Up Arrow (↑)	**Down Arrow (↓)**
BOTTOM	**TOP**
BACKSPACE	

The Left Arrow key (←) moves the cursor one character to the left. If the cursor is at the beginning of a line when you press Left Arrow, the cursor moves to the line terminator at the end of the previous line.

The Right Arrow key (→) moves the cursor one character to the right. If the cursor is positioned on the line terminator at the end of a line when you press Right Arrow, the cursor moves to the first character position on the next line.

The Up Arrow key (↑) moves the cursor up one column position toward the beginning of the buffer. Whenever possible, the cursor stays in the same vertical column. For example, if the cursor is at the third character position on a line when you press Up Arrow, it moves to the third character position on the previous line. If there are less than three characters on the previous line, EDT moves the cursor to the line terminator on the previous line. If you press Up Arrow again and there are three or more characters on the line above, the cursor again moves to the third character position on that new line.

The Down Arrow key (↓) moves the cursor down one column position toward the end of the buffer. Whenever possible, the cursor stays in the same vertical column. For example, if the cursor is at the fifth character position on a line when you press Down Arrow, it moves to the fifth character position on the next line. If there are less than five characters on the next line, EDT moves the cursor to the line terminator on the next line. If you press Down Arrow again and there are five or more characters on the line below, the cursor again moves to the fifth character position on that new line.

Remember that the Up and Down Arrow functions keep track of the character position they started at and use that position if the next line is long enough. However, once you press another key, EDT forgets the position and uses the current cursor position the next time you press either the Up or Down Arrow.

The BACKSPACE function (BACKSPACE; F12 on LK201) does NOT move the cursor back one space. Rather, it moves the cursor to the beginning of the current line. If the cursor is already at the beginning of a line, BACKSPACE moves the cursor to the beginning of the previous line. BACKSPACE always moves the cursor back toward the beginning of the buffer.

The BOTTOM function (GOLD + 4) moves the cursor to the [EOB] mark at the end of the current buffer. Technically, EDT is positioned after the last line terminator in the buffer. If you start adding text when the cursor is on the [EOB] mark, EDT opens up a new line for the text just after the last line. Even though BOTTOM is located on same key as ADVANCE, it has no effect on EDT's direction.

The TOP function (GOLD + 5) is the opposite of BOTTOM. It moves the cursor to the first character position at the top of the buffer. If you start adding text, the new text remains on the first line. The text that was already on the first line is pushed to the right. Although TOP is located on the same key as BACKUP, it has no effect on EDT's current direction.

Note that when you press a key that would move the cursor beyond the beginning of the buffer, EDT prints this message at the bottom of the screen.

```
Backup past top of buffer
```

If you press a key that would move the cursor beyond the end of buffer mark ([EOB]), EDT prints this message at the bottom of the screen.

```
Advance past bottom of buffer
```

In these cases, EDT moves the cursor as far as it can before issuing the message. The cursor will be located either at the beginning or end of the buffer. These messages are warnings. They disappear when a new key is pressed.

The following examples show some of the keys that move the cursor. Consult the key descriptions in Chapter 8 for examples that are not covered in this section. In the text below, the cursor starts on the first character in the paragraph.

```
Progress continues to remain on schedule for the
construction of the new building.  This 250,000
square foot expansion brings the total space to
450,000 square feet.
[EOB]
```

WORD CHNGCASE + WORD CHNGCASE

The cursor is now on the **t** in **to**.

```
Progress continues to remain on schedule for the
```

EOL DEL EOL

The cursor has moved to the position after the **e** in **the** on the first line.

```
Progress continues to remain on schedule for the
```

LINE OPEN LINE + LINE OPEN LINE

The cursor is at the beginning of the third line on the **s** in **square**.

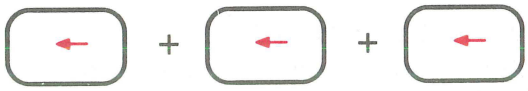
```
Square foot expansion brings the total space to
```

The cursor is now on the **2** in **250,000**. EDT considers the line terminator to be a word, so pressing WORD twice after pressing BACKUP moves the cursor back two words, one of which is the line terminator.

```
construction of the new building. This 250,000
```

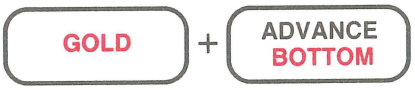
The cursor is now at the beginning of the second line on the **c** in **construction**.

```
construction of the new building. This 250,000
```



The cursor is now on the **h** of **the** at the end of the first line. EDT considers the line terminator to be a character as well as a single-character word.

```
Progress continues to remain on schedule for the
```



The cursor is now on the left bracket of the [EOB] mark at the bottom of the buffer. EDT's current direction is still backward. You need to press ADVANCE to restore the direction to forward.

```
Progress continues to remain on schedule for the
construction of the new building. This 250,000
square foot expansion brings the total space to
450,000 square feet.
[EOB]
```

3.10 Deleting Text

The previous sections showed how to insert text and move the cursor. Next you need to know how to delete text so you can make corrections. Keypad mode has several editing keys that delete text.

DELETE	Delete preceding character
DEL C	Delete current character
DEL W	Delete to end of word
LINEFEED	Delete to beginning of word
DEL L	Delete to next line
DEL EOL	Delete to end of line
CTRL/U	Delete to beginning of line

DELETE and DEL C delete characters. DEL W and LINEFEED delete words or parts of words. DEL L, DEL EOL, and CTRL/U delete lines or parts of lines. For each type of entity (character, word, line) there is an undelete function (UND C, UND W, UND L) that restores the material that you deleted most recently.

3.10.1 Deleting and Undeleting Characters — DEL C, DELETE, UND C

The functions that delete and undelete characters are: DELETE, DEL C, and UND C. These three functions make use of EDT's delete character buffer, one of a group of buffers that you cannot enter or edit and that does not appear in the SHOW BUFFER list. This particular buffer can contain only one character.

You are probably already familiar with the function of the DELETE key (located on the right edge of the main keyboard) because it is used in a number of applications outside EDT. The DELETE key is labeled **DELETE** on most VT100-type terminals as well as on VT52 terminals. On terminals with LK201 keyboards, the DELETE key has the symbol ⟨X⟩ on it.

DELETE has the same basic function in EDT as elsewhere: it deletes the character immediately to the left of the cursor, regardless of EDT's current direction. When the character disappears from the screen, the cursor moves to the left (or up to the end of the previous line, if the deleted character was a line terminator). In EDT, the character has been deleted from your current buffer and placed in the delete character buffer.

You can press the DELETE key as often as you like, continuing to delete character after character from your text. Each time you press DELETE, the contents of the delete character buffer are *replaced* by the new character just deleted.

The DEL C function (⟨,⟩ on VT100; ⟨6⟩ on VT52) also deletes only one character each time you press the key. But the character it deletes is the one that the cursor is on. DEL C is not affected by EDT's current direction.

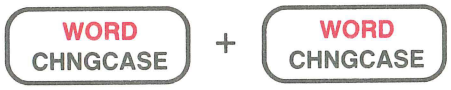
The character deleted by DEL C is also stored in the delete character buffer. Remember, though, that this buffer can only store one character at a time. Each time you use DEL C, you replace the character stored in that buffer. The delete character buffer contains only the most recent character deleted by either DELETE or DEL C.

UND C is the alternate function on the same key as DEL C. You can use UND C to insert the contents of the delete character buffer into your text. When you press GOLD/UND C, EDT copies the character from the delete character buffer to the position just left of the cursor. Because UND C uses two keystrokes to insert a single character, you might not use that function often, but it is useful for transposing letters in a mistyped word.

Here is a line of text as it might appear on your screen. Use DELETE, DEL C, and UND C to correct the errors.

```
Major snow storms in April aer a rare occurrrence.
```

Use the WORD key to move the cursor to the first **s** in **storms**.




```
Major snow storms in April aer a rare occurrrence.
```

Now press the DELETE key to remove the space between **snow** and **storms**.

```
Major snowstorms in April aer a rare occurrrence.
```

Next move to the **e** in the mistyped word **aer**.

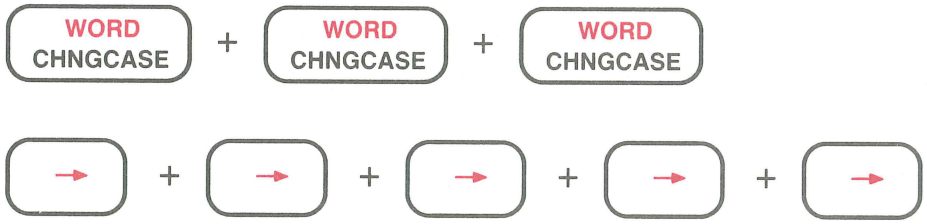
Use DEL C to remove the **e**.

```
Major snowstorms in April ar a rare occurrrence.
```

Now move the cursor so that it is on the space after the **r** and use UND C to put the **e** onto the end of the word.

```
Major snowstorms in April are a rare occurrrence.
```

The last correction is to remove one of the **rs** in **occurrrence**. Move the cursor over to the word you want to fix and then to the appropriate letter. In this case you can remove any one of the three **rs** to make your correction.



The cursor is now on the second of the three **rs**. You can use either DELETE or DEL C to fix the mistake. The end result will be identical.

```
Major snowstorms in April are a rare occurrence.
```

3.10.2 Deleting and Undeleting Words – DEL W, LINEFEED, UND W

The delete word functions are similar to the delete character ones, but involve more characters. There are three functions: DEL W, LINEFEED, and UND W. Note that, as with BACKSPACE, the label on the LINEFEED key has nothing to do with its EDT function. On terminals with LK201 keyboards, the LINEFEED function is assigned to the F13 key.

Word refers to the definition that EDT uses for this particular entity. In EDT, a word is a group of contiguous characters bounded by a space, horizontal tab, line feed, vertical tab, form feed, or line terminator. (A word cannot be split across a line terminator.) For most EDT users, the important boundaries to remember are space and line terminator. (You can change the word boundaries with the SET ENTITY WORD command.)

In deciding which characters make up a word, EDT starts at the current cursor position and moves either backward or forward until it encounters one of the word boundary characters listed above. Except for the space, EDT considers each separating character to be a one-character word itself. EDT generally considers a space to be both a boundary character and part of the word.

EDT uses the delete word buffer with these word functions. Both DEL W and LINEFEED affect the contents of the delete word buffer. This buffer is inaccessible for entering or editing and does not appear on the SHOW BUFFER list. You can use the UND W function to insert the contents of the delete word buffer in your text immediately to the left of the cursor.

The DEL W function does not delete a word in the strictest application of that term. Rather, the function deletes the characters from the cursor position to the end of the word. Therefore, if the cursor is in the middle of a word, only those characters to the right of the cursor, including the cursor character, are deleted. Characters in the word that are to the left of the cursor remain in your text. By contrast, LINEFEED deletes to the beginning of the word and does not include the character that the cursor is on.

Each delete word function can delete a whole word, depending on the cursor location. For DEL W, if the cursor is on the first character of the word, the entire word is deleted and the cursor is now positioned on the first character of the next word. DEL W deletes the space following the word. However, if the word is followed immediately by a line terminator, the line terminator is not deleted.

For LINEFEED, the cursor must be located either on the space immediately following the word or on the first character of the next word for an entire previous word to be deleted. In the first instance, the space following the word is not deleted, so you are left with two adjacent spaces.

UND W inserts the contents of the delete word buffer into your text just to the left of the cursor. If the cursor is at the beginning of a line, the inserted text becomes the first characters on that line.

This two-line example demonstrates the DEL W, LINEFEED, and UND W functions.

```
John, Bill, and Diane went to quarterly meetings in
Jacksonville, Boise, and Denver, respectfully.
```

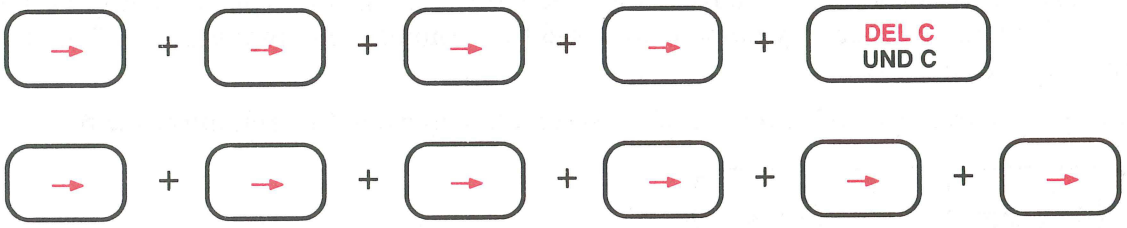
Suppose you want to arrange the names in alphabetical order. Since the cursor is already at **John**, you can delete that name first.

```
Bill, and Diane went to quarterly meetings in
```

Notice that the comma and the space after it are deleted along with the name. Now move the cursor to the **D** in **Diane** and use UND W to insert **John** after the **and**.

```
Bill, and John, Diane went to quarterly meetings in
Jacksonville, Boise, and Denver, respectfully.
```

Since you need to move the comma after **John** to follow **Diane**, move the cursor to the comma, delete it with DEL C, and then put the cursor on the space after **Diane**.

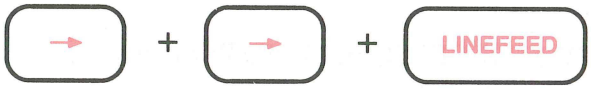



```
Bill, and John Diane went to quarterly meetings in
Jacksonville, Boise, and Denver, respectfully.
```

Use UND C to insert the comma after **Diane**.

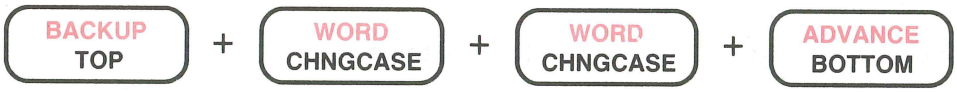
```
Bill, and John Diane, went to quarterly meetings in
Jacksonville, Boise, and Denver, respectfully.
```

Move the cursor two characters to the right to the beginning of **went** and then use LINEFEED to delete **Diane,**(SP).




```
Bill, and John went to quarterly meetings in
Jacksonville, Boise, and Denver, respectfully.
```

Move the cursor back to the **a** in **and** to insert **Diane** in her proper place after **Bill**. By using BACKUP to reverse EDT's direction, you can move back two words. Press ADVANCE before going on with your work so that you do not find yourself in the BACKUP direction for your next editing functions.



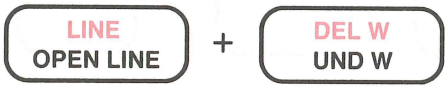

```
Bill, and John went to quarterly meetings in
Jacksonville, Boise, and Denver, respectfully.
```

Now you are ready to use UND W to insert **Diane,**(SP) into the proper location.

```
Bill, Diane, and John went to quarterly meetings in
Jacksonville, Boise, and Denver, respectfully.
```

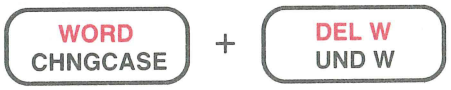
You can use a similar technique to rearrange the cities that these people are going to. This job is a bit easier because each city name is followed by a comma. The final version of the text follows the individual steps.

First move the cursor to the beginning of the second line and delete **Jacksonville,**(SP).



Move the cursor to the **D** in **Denver** and use UND W to insert **Jacksonville,**(SP) into the line.

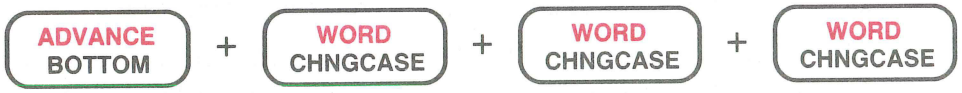
Move the cursor to the **D** in **Denver** and delete **Denver,**(SP).



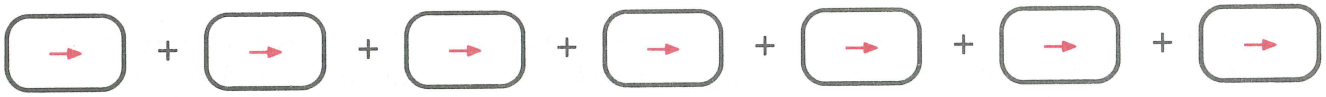
Now backup the cursor to the **a** in **and**; use UND W to insert **Denver,**(SP) into the line.

```
Bill, Diane, and John went to quarterly meetings in
Boise, Denver, and Jacksonville, respectfully.
```

There is one more problem to fix – you need to change **respectfully** to **respectively**. This is a relatively simple operation after those you have just seen. Remember to change EDT's direction to forward. Then move the cursor to the **f** in **respectfully** and delete the remaining part of the word. After that, simply type the necessary letters to correct the typographical error.




```
Bill, Diane, and John went to quarterly meetings in
Boise, Denver, and Jacksonville, respectfully.
```

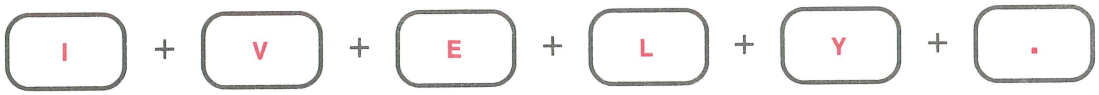



```
Bill, Diane, and John went to quarterly meetings in
Boise, Denver, and Jacksonville, respectfully.
```

Now use DEL W to delete to the end of the word.

```
Bill, Diane, and John went to quarterly meetings in
Boise, Denver, and Jacksonville, respect
```

All that remains is for you to type the correct ending.




```
Bill, Diane, and John went to quarterly meetings in
Boise, Denver, and Jacksonville, respectively.
```

EDT has some SET commands that alter the way EDT uses the word entity. The SET ENTITY WORD command allows you to change the word boundaries for your editing session. The

SHOW ENTITY WORD command displays the word boundaries that EDT is currently using. The SET WORD NODELIMITER command means that EDT will not consider word boundaries as words. See Chapters 6 and 8 for more details on the SET ENTITY, SHOW ENTITY, and SET WORD [NO]DELIMITER commands.

3.10.3 Deleting and Undeleting Lines – DEL L, DEL EOL, CTRL/U, UND L

You can delete a line or part of a line with DEL L, DEL EOL, or CTRL/U. EDT stores the deleted text in the delete line buffer. You can use UND L to insert the deleted line into your current text.

The same concepts apply to deleting lines as do for deleting words or characters. The delete operations work in the same way whether the current editing direction is backward or forward. You can neither enter nor edit the delete line buffer; its name does not appear in the SHOW BUFFER list. The line or line portion that is deleted depends on the location of the cursor when you use a line deleting function.

DEL L and DEL EOL both delete characters to the right of the cursor, including the character the cursor is on. The difference is that DEL L deletes the line terminator at the end of the current line; DEL EOL does not (unless the cursor is positioned on that line terminator).

If the cursor is located at the beginning of a line, DEL L eliminates all characters in the line and places the cursor at the beginning of the next line. If you use DEL EOL instead, EDT deletes all the characters in the line, leaving it blank, with the cursor at the beginning of the blank line. Use DEL L when you want to remove a line from your buffer and go on to the next line. Use DEL EOL when you want to replace the current line with new text.

When the cursor is in the middle of a line, DEL L deletes all the characters to the right of the cursor, including the cursor character, and the line terminator. Nothing is different about this description, but the action on the screen looks very different. Since the line terminator has been removed, the text on the next line moves up to the right of the cursor.

```
This is the first line.
This is the second line.
```

```
This is the This is the second line.
```

You have the beginning portion of the first line, plus the entire second line on a single line. By contrast, when you use DEL EOL in a similar situation, EDT merely removes all the characters to the right of the cursor, including the cursor character, and leaves the next line exactly where it was.

```
This is the first line.
This is the second line.
```

```
This is the
This is the second line.
```

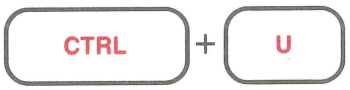
If you press DEL L when you mean to use DEL EOL, simply press RETURN to shift the cursor and the text following it to the next line. (See the descriptions of OPEN LINE and RETURN that appear later in this chapter for information on how EDT adds line terminators to text.)

If you press DEL EOL when the cursor is on a line terminator, EDT deletes the entire following line. On the other hand, if you press DEL L when the cursor is on a line terminator, only that line terminator is deleted.

CTRL/U deletes the text to the left of the cursor to the beginning of the line. CTRL/U has a similar function in other computer programs and utilities, so you might already be familiar with it. In EDT, no line terminator is affected by CTRL/U unless the cursor is at the beginning of a line. When you use CTRL/U with the cursor at the beginning of a line, the entire previous line is deleted.

You can see the effects of CTRL/U with the same sample text.

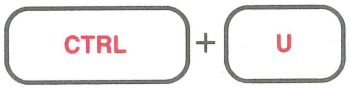
```
This is the first line.
This is the second line.
```




```
first line.
This is the second line.
```

When the cursor is at the beginning of a line, CTRL/U deletes the entire previous line and leaves the cursor in its original position.

```
This is the first line.
This is the second line.
```




```
This is the second line.
```

Each time you use DEL L, DEL EOL, or CTRL/U, the deleted text is moved to the delete line buffer. Only one line or line portion can be stored in the buffer at a time. The contents of the buffer are replaced with new text whenever a line deleting function is used.

The UND L function inserts the deleted line text in your current buffer to the left of the cursor. Try to remember whether or not the deleted text includes a line terminator. When a line terminator is deleted, EDT inserts one when you use UND L. When no line terminator is deleted, EDT does not insert one. If you find that you need to add a line terminator, use the RETURN key. To delete an unwanted line terminator, use DELETE or DEL C.

One of the most common uses for the combination of DEL L and UND L is in reorganizing a list. Delete the line, move the cursor to the line just below the new location, and use UND L to insert the text from the delete line buffer.

This list is arranged alphabetically by state capitals.

```
Augusta, Maine
Boston, Massachusetts
Concord, New Hampshire
Hartford, Connecticut
Montpelier, Vermont
Providence, Rhode Island
```

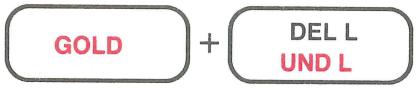
Suppose you need to rearrange the list so the state names are in alphabetical order. In studying the list, you can see that by moving the Hartford line to the top, Maine, Massachusetts, and New Hampshire are in their correct order. Start by moving the cursor to the Hartford line.

```
Augusta, Maine
Boston, Massachusetts
Concord, New Hampshire
Hartford, Connecticut
Montpelier, Vermont
Providence, Rhode Island
```

Now delete the current line.

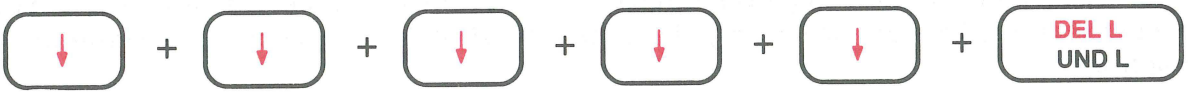
```
Augusta, Maine
Boston, Massachusetts
Concord, New Hampshire
Montpelier, Vermont
Providence, Rhode Island
```

Move the cursor back to the first **A** in **Augusta** and use UND L to insert the deleted line in front of the new current line. Since the deleted line has a line terminator at the end, the inserted text will not push the Augusta line over, but rather form a line above it. The example uses BACKSPACE to move the cursor because it always works backward through the buffer. (You can also reverse EDT's current direction with BACKUP and then press LINE three times. Remember to press ADVANCE to restore the forward direction.)




```
Hartford, Connecticut
Augusta, Maine
Boston, Massachusetts
Concord, New Hampshire
Montpelier, Vermont
Providence, Rhode Island
```

Move the cursor to the **P** in **Providence** and delete that line. Use the down arrow key this time.




```
Hartford, Connecticut
Augusta, Maine
Boston, Massachusetts
Concord, New Hampshire
Montpelier, Vermont
```

Move the cursor up one line and use UND L to insert the deleted line.

```
Hartford, Connecticut
Augusta, Maine
Boston, Massachusetts
Concord, New Hampshire
Providence, Rhode Island
Montpelier, Vermont
```

The finished list has the states in alphabetical order.

DEL EOL is often used when you are composing at the terminal or making changes to the text in your buffer. You might decide that you want to change the ending of the sentence you just inserted. Move the cursor to the character just after the last word you want to keep and use DEL EOL to get rid of the rest of the line. Then you can resume typing.

```
When everyone gets back from vacation, we
can get down to serious business.
```

Using BACKSPACE and WORD, put the cursor at the beginning of **get**. BACKSPACE saves you from having to reverse direction and backup five words. Then use DEL EOL to delete the last five words in the line.

```
When everyone gets back from vacation, we
can
```

Now you can put a new ending on the sentence just by typing the new words.

```
When everyone gets back from vacation, we
can resume work on the library project.
```

You can use DEL EOL and CTRL/U to edit tables. Suppose you have an employee list with department numbers and employee numbers preceding each name. Using CTRL/U, you can delete these numbers and leave only the names. Start with the cursor at the beginning of the first line.

```
14C 974396  Neill Barham
18X 429854  Mona Beckett
12R 423759  Martin Fischer
18X 357984  Debbie Hall
```

Use WORD twice to move the cursor to the first letter in the first name. Then use CTRL/U to delete the number.

```
Neill Barham
18X 429854  Mona Beckett
12R 423759  Martin Fischer
18X 357984  Debbie Hall
```

Now move the cursor down one line and repeat the process.

```
Neill Barham
Mona Beckett
12R 423759  Martin Fischer
18X 357984  Debbie Hall
```

Repeat the key sequences until only the names remain.

```
Neill Barham
Mona Beckett
Martin Fischer
Debbie Hall
```

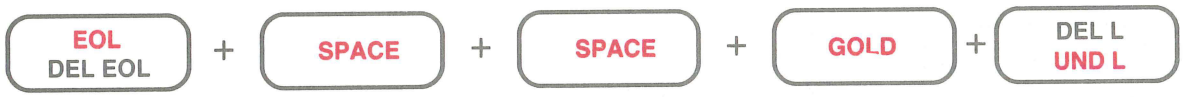
The same sample text can show what happens when you want to invert the order of the words on the line. Suppose you wanted to put the department and employee numbers after the names. The CTRL/U and UND L functions work nicely here. Again, the cursor is at **14C**.

```
14C 974396  Neill Barham
18X 429854  Mona Beckett
12R 423759  Martin Fischer
18X 357984  Debbie Hall
```

Move the cursor to the first letter of the name. Then use CTRL/U to delete to the beginning of the line.

```
Neill Barham
18X 429854  Mona Beckett
12R 423759  Martin Fischer
18X 357984  Debbie Hall
```

Now move the cursor to the end of the line with EOL, add two spaces, and use UND L to insert the deleted numbers.




```
Neill Barham  14C 974396
18X 429854  Mona Beckett
12R 423759  Martin Fischer
18X 357984  Debbie Hall
```

Notice that the cursor is located two spaces after the last number. These are the two spaces that separated the employee number from the name in the original format. Since they will not affect the text, you can ignore them.

Move the cursor to the beginning of the second name on the list and repeat the process until the entire list is changed.

```
Neill Barham  14C 974396
Mona Beckett  18X 429854
Martin Fischer  12R 423759
Debbie Hall  18X 357984
```

3.10.4 Other Uses for the Delete and Undelete Functions

You can use the delete and undelete functions for a variety of purposes. Many times you will find it handy to type some text you want to put in a delete buffer, then delete it, and insert it as needed. For instance, you can create borders with DEL W and UND W. Or, you can use the DEL EOL or CTRL/U functions to insert phrases in your text instead of typing them over and over again. Here are two examples.

Type the following characters:

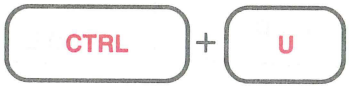
```
/-\-
```

Use LINEFEED to delete them and store them in the delete word buffer. Then use UND W ten times to make a separating line.

```
/-\-/-\-/-\-/-\-/-\-/-\-/-\-/-\-/-\-/-\-/-\-
```

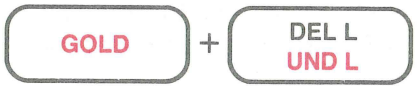
In the next example, you can put the phrase Digital Equipment Corporation in the delete line buffer. Then you can insert the phrase in your text as needed. First type the words, then use CTRL/U to "load" the buffer.

```
Digital Equipment Corporation
```



As long as you do not use any delete line functions that would overwrite the contents of the delete line buffer, you can continue to use UND L to insert the phrase instead of retyping it.

```
Sales representatives from
```




```
Sales representatives from Digital Equipment Corporation
```

3.11 Adding Lines to Your Buffer – RETURN and OPEN LINE

When you are editing or inserting text, you often need to add new lines to the buffer. Several examples have used the RETURN key to do this. Keypad mode also has another function, OPEN LINE, which creates new lines. Neither function is affected by EDT's current direction. The difference between these two functions is the position of the cursor after the key has been pressed.

RETURN uses the RETURN key at the right side of the main keyboard. To use OPEN LINE, press GOLD first and then the 0 (zero) key on the keypad.

Both functions add a line terminator to your text. If you use them when the cursor is at the end of a line, they add a blank line below the current line. If the cursor is in the middle of the line, the text to the right of the cursor shifts to a new line below the current line.

RETURN inserts the line terminator to the left of the current cursor position. The cursor remains on the current character. If the cursor is in the middle or at the end of the line, the cursor and character move to the beginning of the newly created line. If the cursor is at the beginning of the line, EDT leaves the cursor in that position and creates a blank line above the current line. RETURN is generally used to add a new line of text after the end of the current line.

This example uses RETURN to insert a line terminator. Position the cursor at the end of the first line. When you press RETURN, EDT adds a line terminator to your text and positions the cursor at the beginning of the new line.

```
Having a party can be easy if you plan ahead.
You will need to decide what kind of refreshments you
want to serve.
```

```
Having a party can be easy if you plan ahead.

You will need to decide what kind of refreshments you
want to serve.
```

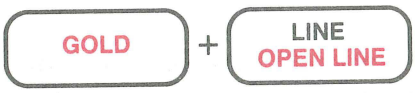
You now have a blank line on which to type your new text.

```
Having a party can be easy if you plan ahead.
First, make a list of people to invite.
You will need to decide what kind of refreshments you
want to serve.
```

OPEN LINE also inserts a line terminator at the current cursor position. However, with OPEN LINE, the cursor is now located on that line terminator. If the cursor is at the end of a line when you use OPEN LINE, it remains at the end of the line; a blank line is created below. If the cursor is in the middle of a line, all the text to the right, including the original cursor character, is moved to the next line, while the cursor remains on the original line. When you use OPEN LINE with the cursor at the beginning of a line, that line is moved down and the cursor is positioned on the newly created blank line.

OPEN LINE is especially useful when you want to insert some text in the middle of the line or above the current line. Simply move the cursor to the character just after where you want to add the text, press GOLD/OPEN LINE, and start typing the new text.

```
A computer depends on people to tell it what to do.
Programmers write programs that tell the computer
how to perform specific jobs.
```




```
A computer depends on people to tell it what to do.
Programmers
write programs that tell the computer
how to perform specific jobs.
```

Now you are ready to type the additional text.

```
A computer depends on people to tell it what to do.
Programmers in offices, laboratories, and factories
write programs that tell the computer
how to perform specific jobs.
```

3.12 Locating Strings in Your Buffer – FIND and FNDNXT

The FIND and FNDNXT (find next) functions move the cursor to new positions in your buffer. But they work differently from the other cursor moving functions such as WORD, LINE, and the arrow keys. They are discussed separately for two reasons: (1) they do not rely on a predefined entity such as character, word, line, or page and (2) they use EDT's search buffer.

Both are located in the same position on the keypads (PF3 for VT100; 8 for VT52). FIND is the alternate function, requiring that the GOLD key be pressed before the FIND key. Both use strings that you supply to determine the new location of the cursor.

Before learning about FIND and FNDNXT, you should remember a few things about the way EDT performs searches. When a search is made, EDT stores the search string characters in its search buffer. This buffer, like the delete character, word, and line buffers, cannot be entered or edited. Each time you enter a string in response to the **Search for:** prompt, EDT overwrites the contents of the search buffer with the new string. Using FIND, you can control the contents of the buffer.

EDT has preset ways of performing searches. The case and diacritical marks of letters are always ignored (thus LEGION, Legion, legion, Légion, and légion are all identical strings for EDT). The cursor is always positioned at the start of the found string. If the search direction is forward, EDT starts the search with the character to the right of the current cursor position and continues to the end of the buffer. If the search direction is backward, EDT starts the search with the character the cursor is on and continues to the beginning of the buffer. (These preset conditions can be changed with the SET SEARCH command, described in detail in Chapter 6.)

3.12.1 FIND

When you use the FIND key, EDT prints a prompt at the bottom of the screen asking you to enter a search string.

```
Search for:
```

For instance, if you want to move the cursor to the word **motorcycle**, type that word after the FIND prompt.

```
Search for: motorcycle
```

Now press the keypad key marked ENTER to send the information to EDT for processing. EDT starts at the current cursor position and searches for the next occurrence of **motorcycle** in your text. If the current direction is forward, EDT searches the text that follows the cursor. Conversely, if the current direction is backward, EDT looks for the string in the text before the cursor.

You can use the ADVANCE and BACKUP functions to complete a FIND operation instead of ENTER. After typing in the string, press ADVANCE to have EDT search in the forward direction. If you want to search for a string that is located before the current cursor position, press BACKUP after typing the string.

When you use ADVANCE or BACKUP, EDT changes its current direction to be that of the editing key you pressed. If you need to find a string toward the beginning of your buffer, press BACKUP after typing the string. But remember that EDT's current direction is now backward. You must press ADVANCE to restore the direction to forward. The next example uses FIND to move the cursor to the words that you want to correct.

```
Sumposiums provide a unique opportunity to meed for a
series of workshopsl, tutorials, and panels.
```

You see the typographical error in the first word of the first line and use FIND to move to that word.

```
Search for:  sump
```

```
Sumposiums provide a unique opportunity to meed for a
series of workshopsl, tutorials, and panels.
```

Notice that you did not need to type the **s** in uppercase. Remember that EDT normally ignores the case of letters in making searches.

Move the cursor to the first **u** in the word and change it to a **y**.

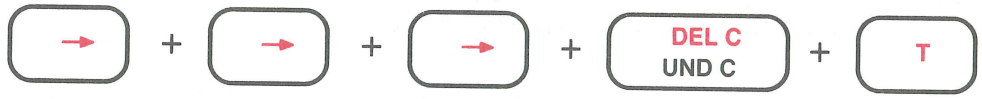
```
Symposiums provide a unique opportunity to meed for a
series of workshopsl, tutorials, and panels.
```

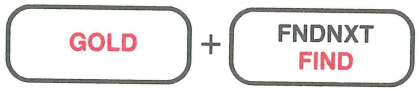
Next you see that **meed** should be **meet**. Use FIND to move the cursor to the mistake. But remember that EDT's current direction is backward, so use ADVANCE to process the search.

```
Search for:  meed
```

```
Symposiums provide a unique opportunity to meed for a
series of workshopsl, tutorials, and panels.
```

First fix that mistake. Then use FIND to move to the last error in the sentence – **workshopsl**. You can use ENTER to process the search because you do not need to change EDT's current direction. EDT does not require that search strings be words or even beginnings of words. In this case, you can type a string that puts the cursor on the mistyped letter. By including the comma with the letter l, you can be more certain of finding the l you want, rather than some other one.






```
Search for:  l,
```

```
Symposiums provide a unique opportunity to meet for a
series of workshopsl, tutorials, and panels.
```

Once you remove the extra letter, the sentence is correct.

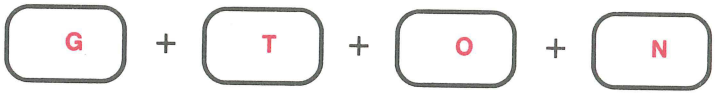
```
Symposiums provide a unique opportunity to meet for a
series of workshops, tutorials, and panels.
```

3.12.2 Editing the Search String or Cancelling the Search

If you make a mistake in typing the search string, you can use the DELETE key to correct it. DELETE is the only function that can edit the search string. Simply press DELETE until you have no characters left after the **Search for:** prompt or until whatever remains to the left of the cursor is correct. Then type the new string or the remainder of the old one.

```
Search for:  Washinton
```

```
Search for:  Washin
```




```
Search for:  Washington
```

If you have typed a search string, but decide to cancel the search operation entirely, use CTRL/U. The cursor returns to its previous position in your text.

3.12.3 FNDNXT (Find Next)

FNDNXT locates the next occurrence of the current search string in your text buffer and moves the cursor to that string. FNDNXT uses EDT's current direction to determine whether to search toward the beginning or end of the buffer. It uses the same search parameters as FIND. The defaults are: case and diacritical marks ignored, cursor placed at beginning of the string, and search continued until the string is found or an end of the buffer is reached.

In order to use FNDNXT, there must be a string in the search buffer. If you try using FNDNXT when the search buffer is empty, EDT moves the cursor to the next line terminator.

When you use FNDNXT, you should be aware of the contents of the search buffer. EDT commands in other modes can affect the text stored in the the search buffer. If you have used one of these commands, EDT might not locate the next occurrence of the string you have in mind.

Several line and nokeypad commands use the search buffer. For example, whenever you use a string for the line mode range specifier, that string is stored in the search buffer. In addition, the line mode substitute commands use the search buffer. In nokeypad mode, the "move" command and the substitute command store strings in the search buffer. Also the nokeypad CLSS command deletes the contents of the search buffer.

3.12.4 Examples Using FIND and FNDNXT

The following examples use both FIND and FNDNXT to locate strings.

```
Palm Beach, Florida
Palmdale, California
Palm Springs, California
Panama City, Florida
```

GOLD + FNDNXT FIND

```
Search for: Flo
```

ENTER SUBS

```
Palm Beach, Florida
Palmdale, California
Palm Springs, California
Panama City, Florida
```

FNDNXT FIND

```
Palm Beach, Florida
Palmdale, California
Palm Springs, California
Panama City, Florida
```

```
Search for: palm
```

```
    Palm Beach, Florida
    Palmdale, California
    Palm Springs, California
    Panama City, Florida
```

```
    Palm Beach, Florida
    Palmdale, California
    Palm Springs, California
    Panama City, Florida
```

3.13 Functions That Manipulate Blocks of Text — SELECT, RESET, CUT, PASTE, APPEND, REPLACE

Keypad mode has several functions that delete and move text:

APPEND	PASTE
CUT	REPLACE

You use the SELECT function to establish a select range so that EDT knows which text you want to delete, move, or copy. The keypad RESET function cancels the select range.

3.13.1 SELECT

There are times when you need to work with a block of text that is neither a character, word, nor line. With a select range, you can perform an editing operation on any group of contiguous characters, such as two or three characters in the middle of a word or two or three paragraphs. SELECT initiates the process of creating a select range.

SELECT (.) marks one end of the group of characters that you want to be a select range. You can include any text entities in a select range. The only restriction is that all the characters must be contiguous; you cannot skip even a single character in creating your range.

To mark a select range, place the cursor at one end of the range, press SELECT, and move the cursor to the other end of the range using any of the cursor moving functions. You can move the cursor forward or backward from the SELECT position. If your select range is to the right of the SELECT position, and you find that you have included too many characters, you can move the cursor to the left to remove excess characters from the end of the range. For example, if you want the select range to have four lines, but you accidentally press Down Arrow five times, simply press Up Arrow once to remove the last line from the select range.

Conversely, if your select range is to the left of the SELECT position, you can move the cursor to the right if you need to remove some characters from that end of the range.

When you use SELECT with a VT100-type terminal, EDT displays the select range text in reverse video so you can clearly see the text you have selected. If you are working with a VT52 terminal, you must remember where the start of the select range is. The other end of the range is always marked by the current cursor position in both types of terminal.

When you mark some text with the SELECT function, EDT considers that text to be an "active" select range. A select range remains active until you use it with a function such as CUT or cancel it with RESET. Use RESET if you find that the select range starts in the wrong place.

3.13.2 Using RESET to Cancel a Select Range

You can use RESET (GOLD + .) to cancel a select range. This is RESET's main function. RESET also resets EDT's direction to forward. If you have created a select range and decide that you no longer need it, press RESET to cancel it. If you have included too many characters in the range, you might find it easier simply to press RESET and start over again, rather than try to move the cursor back to fix the range.

3.13.3 CUT

Select ranges are used most often with the CUT function. When you press CUT (6 on VT100; 3 on VT52), EDT removes the select range text from the current buffer and places it in the PASTE buffer.

The PASTE buffer (see Chapter 2) is automatically created by EDT at the start of every EDT session. Unlike the delete character, delete word, delete line, and search buffers, you can both enter and edit the PASTE buffer, although most of the time you do not need to do either. (You must use a line mode command such as FIND or TYPE to enter the PASTE buffer.)

The PASTE buffer always appears in the SHOW BUFFER list. In fact, EDT does not allow you to delete the PASTE buffer with the line mode CLEAR command; CLEAR only eliminates the contents of the PASTE buffer. (For more information on the CLEAR command, see Chapter 4.)

To use CUT to delete the select range from your text, first create the select range. Then press CUT. The selected text disappears from the current buffer. Whenever you press CUT, EDT replaces the contents of the PASTE buffer with the newly deleted text. When you are simply deleting text, you do not need to be concerned with the contents of the PASTE buffer.

In the following examples, the select range is shaded to simulate the reverse video display on a VT100-type terminal.

```
Meetings this year will be held in these locations:
Boston, MA, Atlanta, GA, Chicago, IL, Cincinnati, OH,
Detroit, MI, and El Paso, TX.
```

SELECT / RESET + WORD / CHNGCASE + WORD / CHNGCASE

```
Meetings this year will be held in these locations:
Boston, MA, Atlanta, GA, Chicago, IL, Cincinnati, OH,
Detroit, MI, and El Paso, TX.
```

CUT / PASTE

```
Meetings this year will be held in these locations:
Boston, MA, Atlanta, GA, Cincinnati, OH,
Detroit, MI, and El Paso, TX.
```

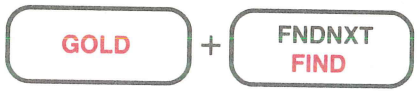
Chicago and **IL,** (as well as the space after each of them) have been deleted. This text is currently stored in the PASTE buffer.

Sometimes you might press CUT when no select range is active either because you cancelled it or never created one. When this occurs, EDT warns you with a message: **No select range active**.

```
Today is Friday the 13th.
```

```
No select range active
```

However, if the cursor is located on the current search string, CUT deletes the search string and moves it to the PASTE buffer.




```
Search for: Friday
```

```
Today is Friday the 13th.
```

```
Today is the 13th.
```

If you already have an active select range when you press SELECT, EDT prints this message:

```
Select range is already active
```

3.13.4 PASTE

To move text from one place to another, use both the CUT and PASTE functions. PASTE (GOLD + 6 on VT100; GOLD + 3 on VT52) inserts the contents of the PASTE buffer into your text to the left of the cursor. Moving text takes these steps:

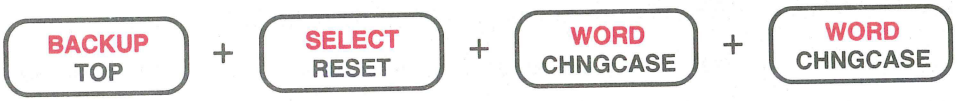
1. Place the cursor at the beginning of the text you want to move.
2. Press SELECT to mark the location.
3. Move the cursor to the end of the text you want to move.

4. Press CUT to delete the text from its current location.
5. Move the cursor to the character just beyond where you want the text inserted.
6. Press GOLD and then PASTE.

Remember that each time you use CUT, the contents of the PASTE buffer are replaced with the newly deleted text.

Using the same sample text as in the CUT example, you can move **Atlanta, GA,** before **Boston** so that the cities are in alphabetical order. This time, reverse EDT's direction and work backward to create the select range.

```
Meetings this year will be held in these locations:
Boston, MA, Atlanta, GA, Cincinnati, OH,
Detroit, MI, and El Paso, TX.
```




```
Meetings this year will be held in these locations:
Boston, MA, Atlanta, GA, Cincinnati, OH,
Detroit, MI, and El Paso, TX.
```

```
Meetings this year will be held in these locations:
Boston, MA, Cincinnati, OH,
Detroit, MI, and El Paso, TX.
```

EDT's direction is still backward, so all you need to do to move the cursor to the **B** in **Boston** is press WORD twice. You might want to press ADVANCE after that to restore the forward direction.

```
Meetings this year will be held in these locations:
Boston, MA, Cincinnati, OH,
Detroit, MI, and El Paso, TX.
```

The final step uses PASTE to put **Atlanta, GA,** at the beginning of the line.

```
Meetings this year will be held in these locations:
Atlanta, GA, Boston, MA, Cincinnati, OH,
Detroit, MI, and El Paso, TX.
```

Notice that **Chicago, IL,** is no longer stored in the PASTE buffer. When you used CUT a second time, the contents of the buffer were replaced by the newly cut text, in this case **Atlanta, GA,**.

You can use CUT and PASTE to copy text that is in one location to another place in your editing session. The difference between moving text and copying it involves only one additional step – between steps 4 and 5 in the list of move operations. The new step uses PASTE to restore the deleted text to its original location before you move the cursor to the new location where the copy will go.

1. Place the cursor at the beginning of the text you want to move.
2. Press SELECT to mark the location.
3. Move the cursor to the end of the text you want to copy.
4. Press CUT to delete the text from its current location.

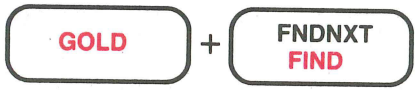
4a. Press GOLD and then PASTE to restore the text to its original location.

5. Move the cursor to the character just beyond where you want the text inserted.
6. Press GOLD and then PASTE.

Suppose you have to update a list of employees and their office locations because some people have moved to a new building. This is a portion of the original list.

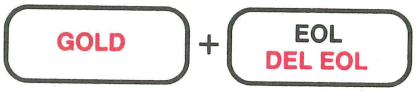
```
Justin Connor      Ross Building
Sarah Balch        Turner Building
Mary West          Turner Building
Evan Schrier       Computer Center
Beth Fifield       Turner Building
```

You need to replace **Turner Building** with **Kingsley Laboratory**. Move the cursor to the first occurrence of **Turner** and use DEL EOL to delete it.




```
Search for: Turner
```

```
Sarah Balch        Turner Building
```




```
Sarah Balch
```

Now type the new location and move the cursor back to the **K** in **Kingsley**.

```
Sarah Balch       Kingsley Laboratory
```

BACKSPACE + WORD CHNGCASE + WORD CHNGCASE

```
Sarah Balch       Kingsley Laboratory
```

Create a select range of **Kingsley Laboratory** and use CUT to move it to the PASTE buffer.

SELECT RESET + EOL DEL EOL

```
Sarah Balch       Kingsley Laboratory
```

CUT PASTE

```
Sarah Balch
```

Use PASTE to restore the deleted text to its location.

GOLD + CUT PASTE

```
Sarah Balch       Kingsley Laboratory
```

Using FNDNXT, move the cursor to the next occurrence of **Turner** and delete the remaining text on the line.

FNDNXT FIND

```
Mary West         Turner Building
```

GOLD + EOL DEL EOL

```
Mary West
```

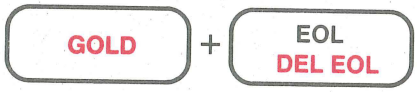
Now use PASTE to add the new location to the end of the line.

GOLD + CUT PASTE

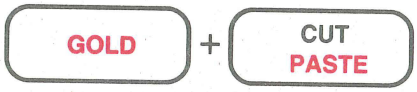
```
Mary West         Kingsley Laboratory
```

Repeat the process for the last occurrence of **Turner Building**.

```
Beth Fifield      Turner Building
```




```
Beth Fifield
```




```
Justin Connor     Ross Building
Sarah Balch       Kingsley Laboratory
Mary West         Kingsley Laboratory
Evan Schrier      Computer Center
Beth Fifield      Kingsley Laboratory
```

3.13.5 APPEND

The APPEND function (9 on VT100; GOLD + ← on VT52) deletes text from your current buffer and places it in the PASTE buffer. When the PASTE buffer is empty, APPEND produces the same results as CUT. The important difference between APPEND and CUT is that APPEND does not overwrite the contents of the PASTE buffer, but rather adds the text it deletes after the last line that is already in PASTE buffer.

Remember that a select range must be a group of contiguous characters. When you use APPEND, you can overcome that restriction. First you can create a select range in one part of your buffer, and use CUT to put the text in the PASTE buffer. Then you can create a second select range at another location in the current buffer, use APPEND to delete that text, and *add* it to the text already in the PASTE buffer. Finally, you can move the cursor to a third location and use PASTE to insert the combined select ranges a the same spot.

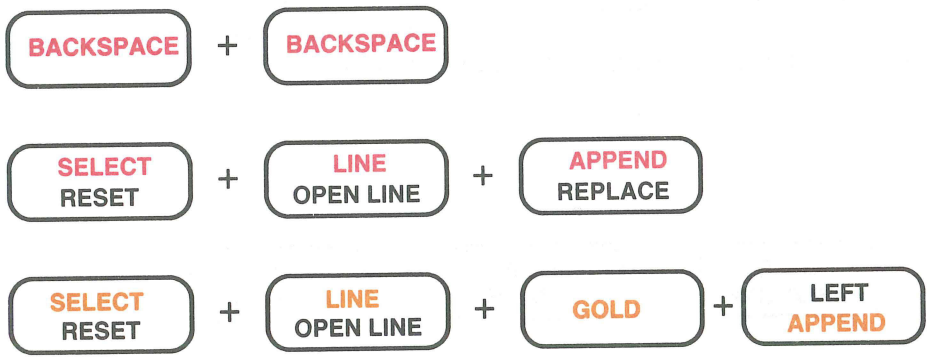
APPEND is useful in rearranging a list.

```
3  COBOL
4  FORTRAN
2  BASIC
5  PASCAL
1  Ada
```

Start with the cursor on the 1. Then create a select range from that line and use CUT to move the line to the PASTE buffer.

```
3  COBOL
4  FORTRAN
2  BASIC
5  PASCAL

```

After you move the cursor to the 2, create a select range from that line and use APPEND to add the line to the PASTE buffer.

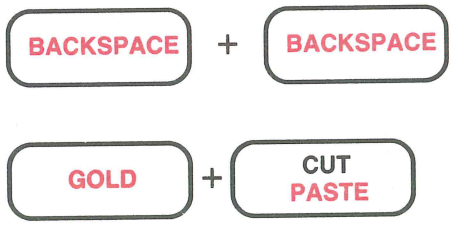



```
3  COBOL
4  FORTRAN
5  PASCAL
```

Notice that lines 3, 4, and 5 are in correct order. All that remains is to insert the contents of the PASTE buffer ahead of line 3. The PASTE buffer looks like this:

```
1  Ada
2  BASIC

[EOB]
```

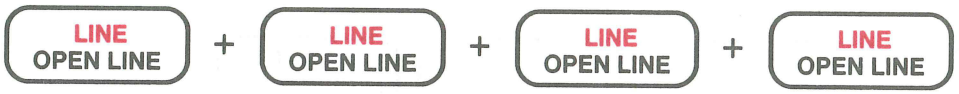
Move the cursor up to the 3 and then use PASTE to insert the two missing lines.




```
1  Ada
2  BASIC
3  COBOL
4  FORTRAN
5  PASCAL
```

APPEND can also help when you find that your original select range did not include all the text you wanted to move or copy. For example, suppose you select a few lines of text and then press CUT to move them into the PASTE buffer. As you start to move the cursor, you realize that you should really add a few more lines. First, use SELECT to mark the additional lines, then APPEND to add them to the lines already in the PASTE buffer.

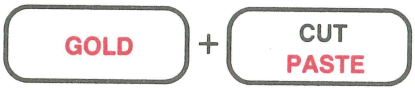
```
As soon as we can make these arrangements, we will
notify your representative.  I expect, however, that
you may have to wait several weeks as our people are
involved in another project.
In the meantime, we hope that a temporary contract
stating each party's intentions will suffice.
```



```
In the meantime, we hope that a temporary contract
stating each party's intentions will suffice.
```

As you get ready to move the cursor, you realize that you wanted to include the remaining two lines with the other text being moved.

After you have moved the cursor to the new location, use PASTE to insert the contents of the PASTE buffer into your current buffer.




```
As soon as we can make these arrangements, we will
notify your representative.  I expect, however, that
you may have to wait several weeks as our people are
involved in another project.
In the meantime, we hope that a temporary contract
stating each party's intentions will suffice.

I will get in touch with you next week to let you
```

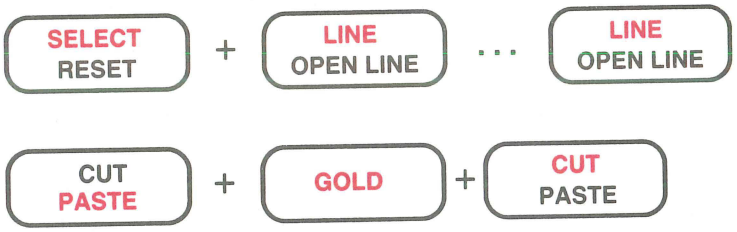
3.13.6 REPLACE

The REPLACE function (GOLD + 9 on VT100; GOLD + ↑ on VT52) allows you to replace the select range text with the contents of the PASTE buffer. After you put the text you want to move or copy into the PASTE buffer, move the cursor to the text you want to replace. Using SELECT, mark the text to be discarded. Then use REPLACE to delete the select range and replace it with the contents of the PASTE buffer.

The following example revises an order notice, announcing a new address. You can use REPLACE to substitute the new address for the old one.

```
The address for placing orders for books has been
changed.  The new address is:

     Book Distribution Department
     Rm. 2005, Smith Building
     35 Anderson Way
     Clifton, NJ 07013
```

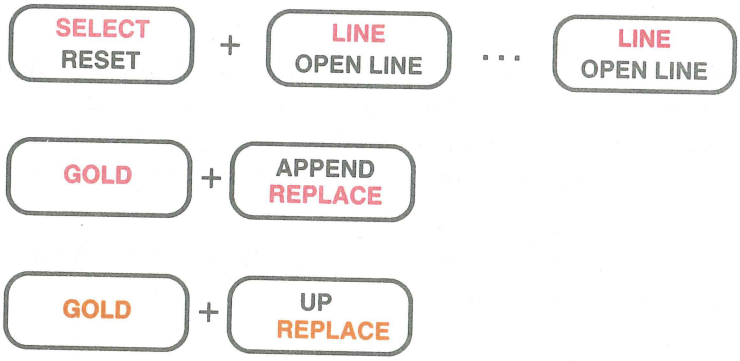
Make the select range include the four lines of the address. Next, use CUT to place a copy of this text in the PASTE buffer. Use PASTE to restore the text to its original location in the memo.




```
The address for placing orders for books has been
changed.  The new address is:

     Book Distribution Department
     Rm. 2005, Smith Building
     35 Anderson Way
     Clifton, NJ 07013

```

Move the cursor to the next location of the old address. Make a select range of the old address and then use REPLACE to enter the new address in that location:

```
Send all orders to:

     Book Distribution Department
     Rm. 103, Jones Building
     2880 Columbia Blvd.
     Clifton, NJ  07013
```




```
Send all orders to:

        Book Distribution Department
        Rm. 2005, Smith Building
        35 Anderson Way
        Clifton, NJ  07013
```

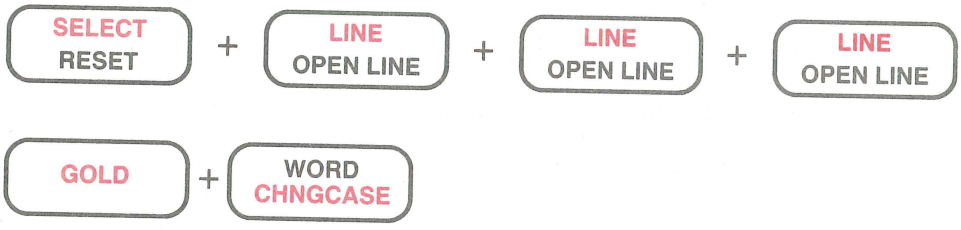
3.14 Other Keypad Functions That Use the Select Range – CHNGCASE and FILL

Select ranges can be used with other keypad functions besides those that move blocks of text between the current buffer and the PASTE buffer. The CHNGCASE function changes the case of all letters in the select range to the opposite case – that is, all uppercase letters are changed to lowercase and all lowercase letters to uppercase. FILL reformats the text in the select range.

3.14.1 CHNGCASE

CHNGCASE (GOLD + 1) changes the case of letters in a select range, the search string, or the character the cursor is on. Whenever there is an active select range, that is, when you have marked a portion of your text by pressing SELECT at one end and moving the cursor to the other, CHNGCASE changes the case of *every* letter in that select range.

```
Gerald Ford
Jimmy Carter
Ronald Reagan
```



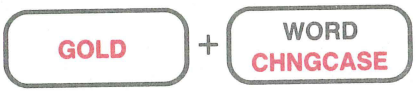

```
gERALD fORD
jIMMY cARTER
rONALD rEAGAN
```

If there is no select range active when you use CHNGCASE, EDT checks to see if the cursor is on the first character of the current search string. If so, CHNGCASE changes the case of every letter in the search string.

```
You can use set commands to change the way EDT operates.
Some operating systems also have set commands.
```

```
Search for:  set
```

```
You can use set commands to change the way EDT operates.
Some operating systems also have set commands.
```

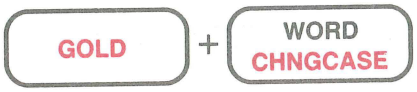



```
You can use SET commands to change the way EDT operates.
Some operating systems also have set commands.
```

```
You can use SET commands to change the way EDT operates.
Some operating systems also have SET commands.
```

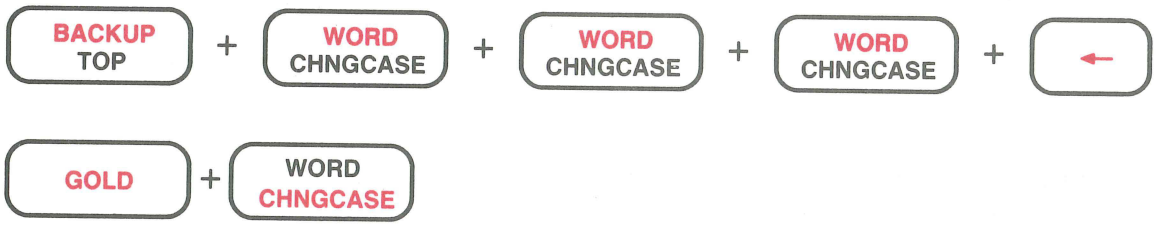
Most of the time when you use CHNGCASE, there will be no active select range and the cursor will not be positioned at the beginning of the current search string. In these instances, CHNGCASE merely changes the case of the letter that the cursor is on if EDT's current direction is forward or the letter to the left of the cursor if the current direction is backward.

```
NIh is a united States government agency.
```




```
NIh is a United States government agency.
```

Now you notice that the **h** is lowercase in **NIh**. Reverse EDT's direction, move the cursor back to the space just after the **h**, and use CHNGCASE.




```
NIH is a United States government agency.
```

3.14.2 FILL

The FILL function is available on the VT100 keypad as GOLD + 8. If you are using a VT52 terminal, EDT uses the predefined key sequence CTRL/F to perform this function.

When you type and edit text with EDT, the length of your text lines can become ragged. Some lines are short; others are long. Filling text enables you to even out the line lengths and improve the appearance of your work. (If you use a text formatter such as RUNOFF, your formatter takes care of uneven line lengths, so you do not need to use EDT's FILL function.)

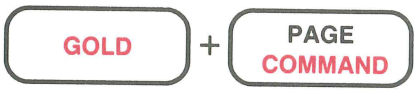
In keypad editing, FILL reformats the text in the select range. It uses the current SET WRAP value to determine how many whole words to put on a line. If SET NOWRAP (the default) is in effect, EDT uses a line length of one less than the current SET SCREEN width value. The default screen width is generally 80 characters, thus by default, the FILL line length is 79. (See Chapter 8 for details on SET WRAP and SET SCREEN.)

In filling your text, EDT puts as many whole words on a line as it can without causing a break in a word and without exceeding the current line length. Line terminators are replaced with spaces and new line terminators are added in place of spaces where necessary.

In the following example, the COMMAND function (GOLD + 7) accesses the line mode SET WRAP command. When SET WRAP is in effect, EDT uses the SET WRAP value as the line length for the FILL function. When you are inserting text in keypad mode, the SET WRAP value also determines the maximum line length. To cancel SET WRAP, issue the SET NOWRAP command.

To reformat this sample text, first establish a SET WRAP value of 50. Create a select range from your text, and then use FILL to reformat the lines. After the text has been filled, you can issue the SET NOWRAP command to return EDT to the default NOWRAP state.

```
I happen to be tied up in a budget meeting all day
on the 31st.
Otherwise, I would attend the Utilities Program
Team meeting.
In the meantime, I am sending you a description
of our requirements for the new product.
```



```
Command: SET WRAP 50
```

ENTER SUBS

SELECT RESET + LINE OPEN LINE ... LINE OPEN LINE

GOLD + SECT FILL

CTRL/F

```
I happen to be tied up in a budget meeting all day
on the 31st. Otherwise, I would attend the
Utilities Program Team meeting. In the meantime, I
am sending you a description of our requirements
for the new product.
```

GOLD + PAGE COMMAND

```
Command: SET NOWRAP
```

ENTER SUBS

3.15 Substituting in Keypad Mode – SUBS

The SUBS function (GOLD + ENTER) makes use of both the search buffer and the PASTE buffer. When a substitution function is performed by a computer editor, it usually involves replacing one string with another string. When you use the keypad SUBS function, the string you want to find and delete must be contained in EDT's search buffer. The replacement or substitute string must be located in the PASTE buffer.

The easiest way to "load" these buffers is with FIND and CUT. Use FIND to put the first string in the search buffer. Then type the substitute string in a convenient location in your current buffer and use CUT to remove it to the PASTE buffer. You can load the buffers in either order.

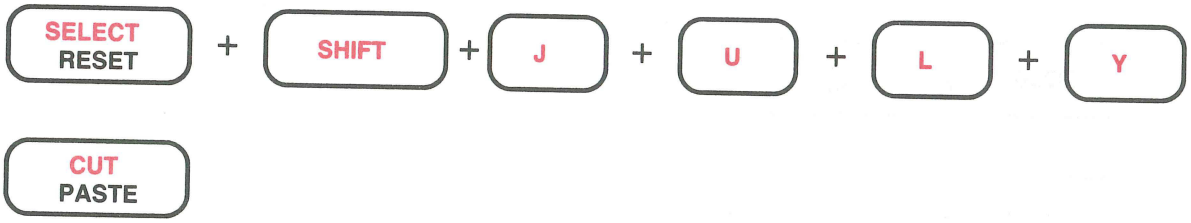
When you use SUBS, EDT performs the substitution first and then moves to the next occurrence of the search string. Thus, you want to be sure that the cursor is at a search string match when you press SUBS. If EDT cannot find another occurrence of the search string in the current direction, it prints the message **String was not found**.

You can use SUBS repeatedly to replace all the first strings in a buffer with the second. After the last substitution is made, the warning message appears.

This example replaces all occurrences of **June** with **July**.

```
In our June report, we discussed the results
of the June sales initiative.  The June sales
figures exceeded our predictions by 20%.  Sales
exceeded those of the previous June by 35%.
```

The first thing to do is put **July** in the PASTE buffer.



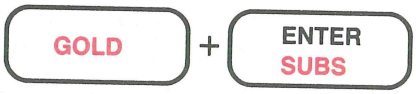
The next step is to load the search buffer. Notice that you do not have to capitalize the **j** in **june** because EDT generally ignores case when making searches.

```
Search for: june
```

```
In our June report, we discussed the results
```

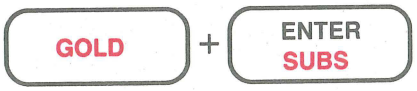
Now you are ready for the SUBS function.

```
In our July report, we discussed the results
of the June sales initiative.  The June sales
figures exceeded our predictions by 20%.  Sales
exceeded those of the previous June by 35%.
```




```
In our July report, we discussed the results
of the July sales initiative.  The June sales
figures exceeded our predictions by 20%.  Sales
exceeded those of the previous June by 35%.
```

```
In our July report, we discussed the results
of the July sales initiative.  The July sales
figures exceeded our predictions by 20%.  Sales
exceeded those of the previous June by 35%.
```




```
In our July report, we discussed the results
of the July sales initiative.  The July sales
figures exceeded our predictions by 20%.  Sales
exceeded those of the previous July by 35%.

String was not found
```

Notice that EDT prints the message **String was not found** when it finishes the last substitution. Remember that EDT first substitutes and then searches. After it makes the last substitution, it still has to try to find the search string again.

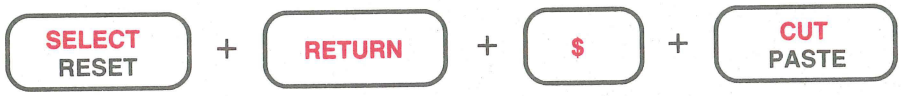
The fact that EDT performs the search after the substitution allows you to decide whether you want to make that particular substitution. If you do not, simply press FNDNXT to move the cursor to the next occurrence of the search string. If you want to change that one, use SUBS again. Thus, you can move through the buffer using SUBS each time you want to make a replacement and FNDNXT when you want EDT to leave the search string alone, but still move on to the next occurrence.

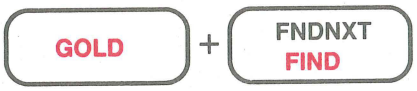
If the cursor is not located at the search string when you press GOLD/SUBS, EDT prints the message **No select range active.** Pressing FNDNXT takes care of the situation.

You can use SUBS to alter the beginnings or ends of lines. Use the carriage return character (CTRL/M) to stand for the line terminator in both the search and substitute strings. Suppose you had a list of prices and needed to add a dollar sign in front of each one:

```
129.95
 38.50
154.00
 79.99
```

You can use RETURN to put the carriage return character in each buffer. Notice that EDT prints **^M** for the line terminator symbol.






```
Search for: ^M
```

(Press RETURN here; do not type ^M.)

Both the search and PASTE buffers are loaded. Move the cursor back to its initial position above the 1 in the first line. Now you are ready for SUBS.

```
$129.95
$ 38.50
$154.00
$ 79.99
```

3.16 Using EDT's Preset Tab Stops – TAB

The TAB key, located near the upper left corner of the main keyboard, shifts text to the right of the cursor position to EDT's preset tab stops. The cursor character is also shifted, so that the cursor moves to the right, but remains on the same character where it was before TAB was pressed. TAB always moves text to the right regardless of EDT's current direction.

EDT has a preset tab stop every eight character positions. Each time you press the TAB key, EDT shifts the text starting at the current cursor position to the next preset tab stop. This means that if the cursor is positioned anywhere before the ninth column on the screen when you press TAB, the text is moved to column 9. Similarly, if the cursor is located anywhere from column 9 to 16, pressing TAB moves the cursor character to column 17.

The following example shows what happens when you use the TAB key with EDT's tab default – SET NOTAB – in effect.

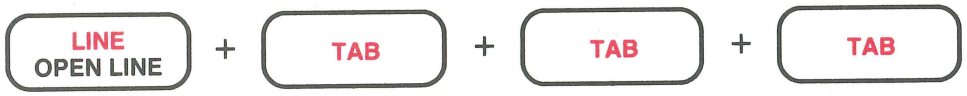
```
This is the first line.
This is the second line.
This is the third line.
This is the fourth line.
```

```
This is the first line.
        This is the second line.
This is the third line.
This is the fourth line.
```

Now move the cursor to the beginning of the third line.

```
This is the first line.
        This is the second line.
                This is the third line.
This is the fourth line.
```

Finally, move the cursor to the beginning of the fourth line.




```
This is the first line.
        This is the second line.
                This is the third line.
                        This is the fourth line.
```

When EDT moves text a full tab stop, it does not insert eight spaces in your text. Rather, it inserts a horizontal tab character (CTRL/I) in the buffer. If you want to delete the horizontal tab character, simply press DELETE once and the text shifts back to the left eight spaces.

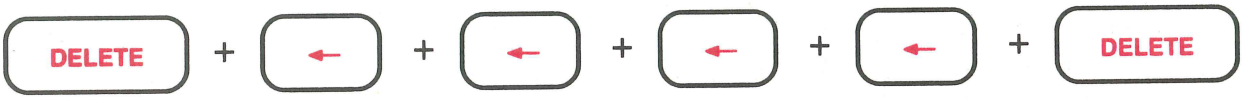
```
This is the first line.
        This is the second line.
```

```
This is the first line.
This is the second line.
```

The next example uses TAB to move text less than eight spaces to the nearest tab stops. Then you can use DELETE to put the text back in its original format.

```
JAN FEB MAR
```

```
JAN     FEB     MAR
```




```
JAN FEB MAR
```

Keypad mode has other tabbing functions that enable you to indent lines of text. They are: CTRL/A, CTRL/D, CTRL/E, and CTRL/T. CTRL/T enables you to indent a block of text. None of these tabbing functions does anything to your text unless a SET TAB value has been established for the editing session. A complete explanation of these functions appears in the tabbing section in Chapter 7.

3.17 Inserting Special Characters in Your Text — SPECINS (Special Insert)

The SPECINS function (GOLD + 3 on VT100; GOLD + → on VT52) enables you to insert any character from the DEC Multinational Character Set into your text. These characters include all the ASCII control characters (decimal values 0 through 31), DEL/RUBOUT (decimal 127),

and the special characters in the DEC Multinational Character Set (decimal 128 through 255). With SPECINS, you must use the decimal equivalent value of the character you want to insert. See Appendix G for a list of the characters in the DEC Multinational Character Set and their decimal equivalent values.

Control characters can be used in EDT startup command files and macros. (See Chapter 7 for details on these facilities.) You can also use control characters in programs to control various features of the computer system.

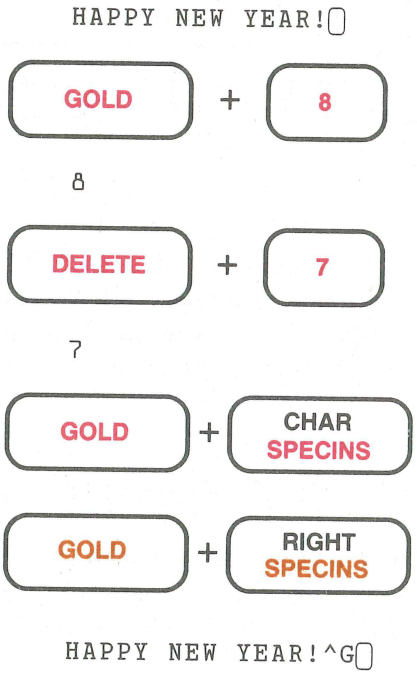
The special characters in the DEC Multinational Character Set include letters with various diacritical marks for use with languages other than English. SPECINS enables you to insert these characters in your text, even if your terminal cannot display the characters.

There are four steps involved in using the SPECINS function. Each time you need to enter a special character you must repeat these four steps, because EDT can process only one decimal value at a time.

1. Press GOLD.
2. Type the decimal equivalent of the special character you want to insert in the text. Be sure to use the keyboard number keys, not the keypad ones.
3. Press GOLD again, this time to access the alternate keypad function.
4. Press the SPECINS keypad key.

EDT uses symbols to represent the special characters in the DEC Multinational Character Set. The symbol for most control characters consists of the circumflex (^) preceding the control character, for example, ^G for CTRL/G. If your terminal has eightbit character display, you will see Á for decimal character 193. For terminals without eightbit character display, EDT uses the symbol <A'> to display character 193 on the screen.

When you press the keyboard digit(s) after GOLD, EDT displays the number at the bottom of your screen. You can use the DELETE key to edit the number if you make a mistake. After you have entered the decimal number correctly, press GOLD/SPECINS to insert the character in your text.



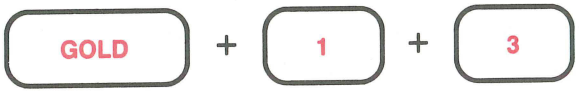
EDT displays ^G when it reads the text during your editing session; it does not sound the bell. However, when you display the text on your terminal under the operating system control, the terminal bell sounds when the system sends the CTRL/G character to your terminal.

To delete a control symbol, you need to use only one delete character function to remove both the circumflex (^) and the character.

EDT represents some other control characters by abbreviations inside angle brackets, for example <FF> for form feed (CTRL/L). As with the circumflex/character combination, only one delete character function is needed to remove the entire symbol.

You can use SPECINS to enter a carriage return character in a startup command file or macro.

```
SET ENTITY WORD " □
```




```
13
```

```
SET ENTITY WORD " <CR>□
```

Note that if you are working in an EDT environment with SET NOREPEAT in effect, you cannot use the SPECINS function. However, the default for EDT is SET REPEAT.

3.18 The Keypad Repeat and Backward Functions — GOLD + [–][n]

In keypad mode, there are times when you want to repeat a function several times, for instance when you are moving the cursor, deleting something, or undeleting. Instead of pressing that editing key or key sequence over and over again, you can use the GOLD repeat feature. The repeat feature enables you to repeat a function up to 32767 times.

Some functions allow you to use GOLD with the minus sign (–) to temporarily override EDT's current direction when that direction is forward. You can use this feature to backup for a single function or a repeated function.

Not all keypad functions lend themselves to being repeated. For example, you cannot repeat COMMAND or RESET. Functions that are not sensitive to EDT's direction are not affected by a minus sign.

These functions can be used with the repeat feature. The asterisk marks functions that you can precede with a minus sign to make them work backward without changing EDT's forward direction.

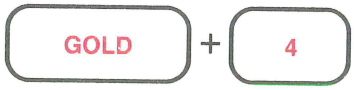
BACKSPACE	FIND*	SECT
CHAR*	FNDNXT*	SUBS
CHNGCASE*	LEFT ARROW	TAB
DEL C	LINE*	UND C
DEL EOL	LINEFEED	UND L
DEL L	OPEN LINE	UND W
DEL W	PAGE*	UP ARROW
DOWN ARROW	PASTE	WORD*
EOL*	RIGHT ARROW	

To repeat a function, first press GOLD and then type the number of repeats that you want. You must use the keyboard number keys to enter the repeat number, not the keypad ones.

When you type the repeat number, EDT displays it below your text. You can edit the repeat number with the DELETE key. Notice that DELETE is not among the functions that can be repeated. If it were, you would have no way to edit the repeat number.

Suppose you need to move the cursor forward four words. Press the following keys:

```
This line has five words.
```

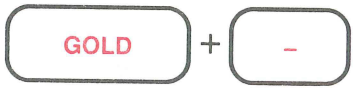



```
4
```

```
This line has five words.
```

When you use the minus sign, EDT reverses direction for that one function only if the current direction is forward. The minus sign precedes the repeat count. If you only need to back up once, simply type the minus sign after pressing GOLD; then press the editing key.

```
This is the first line.
This is the second line.
```



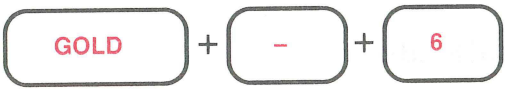

```
-
```

```
This is the first line.
This is the second line.
```

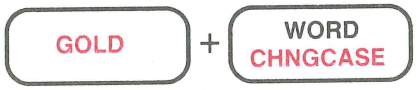
When you use the repeat feature with functions that affect the contents of the delete character, word, or line buffer, only the *last* deletion remains in the buffer. Thus, if you delete six characters, only one character is in the delete character buffer – the last character that EDT deleted.

In the next example, move the cursor backward using the minus sign to change the case of six letters.

```
EDT is the Digital standard editor.
```




```
-6
```




```
EDT is the DIGITAL standard editor.
```

If you change your mind about using the repeat function, and have only pressed GOLD and the number, either delete the entire repeat number or press CTRL/U and the proceed with your editing.

Note that if you are working in an EDT environment with SET NOREPEAT in effect, you cannot use the GOLD repeat feature. However, the default for EDT is SET REPEAT.

3.19 Using Commands from Other EDT Modes — COMMAND, CTRL/Z

If you have read everything in this chapter so far, you have seen that keypad editing contains a wide variety of functions. However, EDT has other functions available in the other editing modes that you might want to use from time to time. There are ways to switch back and forth between the editing modes, as well as ways to enter commands from other modes without leaving keypad mode.

All these possibilities are described in the section on Accessing Other Modes in Chapter 7. This section explains how to use the keypad mode COMMAND function to access a line mode command directly from keypad mode and how to move from keypad mode to line mode.

There are a number of line mode commands that you will find useful when editing in keypad mode. Some of these are:

FIND – can be used to move EDT to another buffer.

CLEAR – deletes a buffer and its contents.

DELETE – can be useful in deleting large blocks of text.

SUBSTITUTE – can perform substitutions throughout the entire buffer. Does not require loading the PASTE buffer.

SUBSTITUTE NEXT – performs a search and substitution without involving the PASTE buffer.

SET – allows changing various EDT default characteristics.

SHOW – enables you to find out information about your editing session.

INCLUDE – puts a copy of an external file into your text.

WRITE – puts a copy of part or all of a buffer into an external file.

The SET and SHOW commands are described in Chapter 8. A summary table of them is given in Chapter 6 along with a detailed discussion of some of the more frequently used commands. Information about other line mode commands appears in Chapters 4 and 8.

3.19.1 Using Line Mode Commands from Keypad Mode

You have already seen how the keypad mode COMMAND function (GOLD + 7) enables you to use the line mode EXIT or QUIT command directly from keypad mode. You can use COMMAND to access any line mode command without having to leave keypad mode.

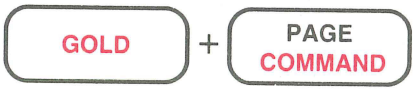
To use the COMMAND function:

1. Press GOLD + COMMAND.
2. After EDT displays the **Command:** prompt below your text, type the line mode command you want EDT to perform.
3. Press ENTER (not RETURN) to signal EDT to process the command.

You can edit your command if you change your mind or have made a mistake, but only with the DELETE key. If you decide not to issue a line mode command, you can use CTRL/U to return the cursor to the text buffer.

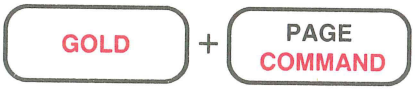
If you accidentally press RETURN to process the line mode command, EDT prints the RETURN symbol ^M and waits for you to press another key. Simply use DELETE to remove the ^M (only press DELETE once) and then press ENTER.

This example uses the line mode FIND command to shift EDT from the current buffer to the top of the PASTE buffer.




```
Command:  FIND =PASTE
```

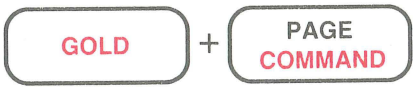
In the next example, you can type the line mode SHOW BUFFER command to find out which buffer you are in. The equal sign (=) points to the current buffer.




```
Command: SHOW BUFFER
```

```
Command: SHOW BUFFER
 MAIN    2796      lines
=PASTE   7         lines
Press return to continue
```

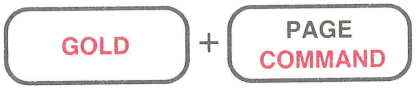
The following example deletes all the lines from the current cursor line to the end of the buffer. You do not need to know how many lines you plan to delete or what the EDT line numbers are for those lines.




```
Command: DELETE REST
```

```
Command: DELETE REST
37 lines deleted
```

This last example substitutes **BASIC-PLUS-2** for every instance of **BASIC-PLUS**.




```
Command:  SUBSTITUTE!BASIC-PLUS!BASIC-PLUS-2! WHOLE /NOTYPE
```

```
Command:  SUBSTITUTE!BASIC-PLUS!BASIC-PLUS-2! WHOLE /NOTYPE
16 substitutions
```

Notice the exclamation points. These are the delimiters surrounding the search and substitute strings. The slash preceding NOTYPE marks that word as a qualifier for the SUBSTITUTE command.

3.19.2 Shifting to Line Mode for a Series of Commands

When you need to enter several line mode commands at one time, or use the line mode HELP facility, you might find it more convenient to shift to line mode, process all your commands, and then return to keypad editing or end your EDT session.

CTRL/Z performs the shift from keypad mode to line mode. As soon as you press CTRL/Z, EDT displays the line mode asterisk prompt (*) at the left of your screen below your text. When you see the asterisk, you can enter a line mode command.

Issue as many line mode commands as you need. Then, if you are finished with your editing work, type the line mode EXIT or QUIT command to leave EDT. If you want to do more editing in keypad mode, use the line mode CHANGE command to return to keypad editing. EDT shifts to keypad mode and displays the text that surrounds the current line. You can tell EDT to move to a different portion of your text by including a range specifier with the CHANGE command. See Chapter 8 for information on the CHANGE command. Details on the line mode range specifier appear in Chapter 4.

The following example assumes that you have just typed a list of categories for a data sheet. You want to make five more copies of this list and place the entire text in a different buffer. Since this work involves a number of line mode commands, you press CTRL/Z when you finish typing the text to shift EDT to line mode.

```
Name:
Title:
Addr-1:
Addr-2:
City:
State:
ZIP:
AC:
Phone:
```

```
*
```

Now you are ready to issue the line mode commands.

```
*COPY WHOLE TO END /DUPLICATE:5
10 lines copied 5 times
```

There are now 60 lines in your buffer, with a total of six copies of the list. Next you want to put a copy of the entire buffer in an external file so you can use it at a later date. The WRITE command creates the file called ADDRFORM.DAT and puts a copy of the entire current buffer in that file.

```
*WRITE ADDRFORM.DAT
DISK$USER:[SMITH]ADDRFORM.DAT;1 60 lines
```

Now you want to put a copy of the data sheet in another buffer so you can have it available after you fill in the contents of the first page. The COPY command copies the whole buffer to another buffer called FORM. The COPY command creates the FORM buffer.

```
*COPY WHOLE TO =FORM
60 lines copied
```

When you use the COPY command to copy text to another buffer, EDT is transferred to that buffer. At the end of the COPY command, you are in the FORM buffer. To return to the MAIN buffer where you want to work, use the FIND command. Then use the CHANGE command to resume keypad editing.

```
*FIND =MAIN
*CHANGE
```

If you only want to create the data sheet and save a copy of it, all you need to use is the COPY command and then the EXIT command.

```
*COPY WHOLE TO END /DUPLICATE:5
10 lines copied 5 times

*EXIT
DISK$USER:[SMITH]DATASHEET.DAT;1 60 lines
```

See Chapter 4 for a detailed explanation of the line mode commands shown in these examples.

Chapter 4
Line Editing

4.1 Introduction

The focus of this chapter is on editing text files with line mode commands. Commands that modify EDT's behavior and those that deal with EDT's special features are discussed later. Information on the SET and SHOW commands appears in Chapters 6 and 8. Details on DEFINE KEY, DEFINE MACRO, and TAB ADJUST are given in Chapter 7. A complete description of CHANGE appears in Chapters 5 and 8.

You can use EDT's line editing facility with any interactive terminal – hardcopy or screen. You can perform all functions necessary to edit text, as well as functions that create or insert external files during your editing session. You can use line mode commands directly from both screen modes to extend the capabilities of keypad or nokeypad editing.

Line editing uses the line as its point of reference. In line mode, EDT moves through the text in a buffer line by line, not character by character as in EDT's other editing modes. EDT uses a system of line numbers to identify each line in the text uniquely. EDT maintains these line numbers even if you are using one of the screen modes and never have occasion to see them.

Line mode commands carry out a variety of functions. Some commands perform editing operations; other commands affect your session.

EXIT **QUIT**	End your EDT session.
HELP	Provides online information about EDT during your editing session.
INSERT	Enables you to add text during your editing session.
DELETE	Deletes whole lines from an EDT buffer.
FIND	Moves EDT to a new location in your editing session.
TYPE <**null**>	Move EDT to a new location in your editing session and display that text.
SUBSTITUTE **SUBSTITUTE NEXT**	Delete a string and replace it with another string.
MOVE	Moves one or more lines from one location to another.
COPY	Puts a copy of one or more lines in a second location.
REPLACE	Deletes one or more lines and enables you to insert new text in that location.

CLEAR	Deletes a buffer.
INCLUDE	Copies an external file into your EDT session.
WRITE **PRINT**	Put a copy of one or more lines in an external file.
RESEQUENCE	Renumbers the EDT line numbers.
FILL **TAB ADJUST**	Reformat one or more lines of text.
CHANGE	Shifts EDT to a change mode.
SET	Changes the way EDT operates during your editing session.
SHOW	Indicates various types of information about your editing session.
DEFINE KEY	Enables you to extend keypad mode editing functions through EDT's key definition facility.
DEFINE MACRO	Enables you to extend line mode editing commands through EDT's macro facility.

4.2 Definitions of Terms Used in Chapter 4

These terms are used in describing line editing. Definitions of other EDT terms appear in Chapter 2.

contiguous lines

A group of adjacent lines, whose line numbers form a sequence that does not omit any existing line number.

delimiter

A punctuation character used to set off strings in commands. In certain cases, strings require quotation marks (either single – ', or double – ") as delimiters.

range

A reference to one or more lines in a text buffer.

syntax

The structure of an EDT command statement.

4.3 Using Commands in Line Editing

Line mode uses the asterisk (*) as its prompt character. Whenever EDT displays the asterisk at the left of your screen or paper, you are in line mode and ready to issue a line mode command.

Line mode commands are composed of one or more of these elements: command words, specifiers, and qualifiers. All commands except the <null> command use a command word or phrase, for example, INSERT, COPY TO, or SUBSTITUTE NEXT. Many command words can be abbreviated. This manual shows the abbreviation by underlining the required letters in the command syntax statements both in this chapter and Chapter 8. Commands are always spelled out fully in the text and examples.

A typical line mode command syntax statement looks like this:

```
COMMAND WORD [specifier] [specifier] [/QUALIFIER]
```

In the following example, the minimum abbreviations for both the command word DELETE and the qualifier /QUERY are indicated by underlining the abbreviation:

```
DELETE [=buffer] [range] [/QUERY]
```

Some command words and qualifiers require more than a single letter for their minimum abbreviations. Look for the underlined letters in each command syntax statement to find out the abbreviation for that command and for its qualifiers if it has any.

4.3.1 Specifiers

Specifiers are pieces of information that you supply to EDT to complete a command statement. Most specifiers tell EDT what part of the text you want the command to affect. Many commands require that you supply at least one specifier with the command. The most frequently used line mode specifiers are:

```
buffer
file-specification
range
string
```

In command syntax statements, specifiers are typed in lowercase letters to indicate that you must supply information of the sort indicated by the specifier word(s). For example, when you see the word **buffer** in a syntax statement, replace it with the name of a buffer, such as PASTE.

When you see a specifier enclosed in square brackets ([ ]) in a syntax statement, it means the specifier is optional. If you omit an optional specifier, EDT generally uses a default value. For instance, if you do not include a buffer specifier with a DELETE command, EDT assumes that you want to delete text in the current buffer.

4.3.2 Qualifiers

Qualifiers modify the way in which the command is carried out. They are always optional. Qualifiers differ for each line mode command. Some commands take none; others can take as many as three. Like command words, qualifiers can be abbreviated. The slash distinguishes a qualifier from a command word. Whenever you use a qualifier with a line mode command, you must precede the qualifier with a slash (for example, /SAVE). Qualifiers are always the last element in a line mode command line.

Line mode qualifiers are:

```
/BRIEF                  /SAVE
/DUPLICATE              /SEQUENCE
/NOTYPE                 /STAY
/QUERY
```

The section, Qualifiers in Line Mode, which appears later in this chapter, has more information on qualifiers.

4.4 Buffer and Range Specifiers

Most line mode commands require you to tell EDT which line, group of lines, or buffer to work on. You use the range specifier to indicate which line or group of lines you want the command to affect. Use the buffer specifier to indicate the appropriate buffer. If you do not supply one of these specifiers, EDT assumes the current buffer for buffer. The default range can be the current line, the first line of a buffer, or the entire contents of a buffer.

The term location refers to the combination of a buffer and range. The line mode commands MOVE and COPY involve two locations. Location-1 refers to the lines you want to move or copy. Location-2 refers to the place where you want to put the text. For these commands the specifiers are buffer-1, buffer-2, range-1, and range-2.

4.4.1 The Buffer Specifier

You can use buffers during your EDT session to store different pieces of text. When you want to move from one buffer to another you need to include the buffer specifier in your line mode command. In most instances, EDT moves to the buffer that you specify. The PRINT and WRITE commands leave EDT in the current location, regardless of whether you specify a buffer name with the command. CLEAR also does not move EDT to the named buffer. However, if you are using CLEAR to delete the current buffer and the current buffer is not MAIN, EDT moves to the MAIN buffer. (See Chapter 2 for more general information on buffers.)

EDT has two signals for the buffer specifier: the word BUFFER and the equal sign (=). All line mode commands except CLEAR require that one of these signals precede the buffer name. There is no space between the equal sign signal and the buffer name; there is a space between the word BUFFER and the buffer name.

```
*FIND BUFFER PASTE
*FIND =PASTE
```

The examples in this manual use the equal sign indicator.

Buffer names must begin with a letter. Names can include any alphanumeric characters (letters and digits) and the underscore (_). You can use as many as 80 characters in a buffer name. However, since you will have to type the buffer name at least once, you will probably want to avoid long buffer names.

When you use a range specifier with a buffer name to indicate a line or group of lines in that buffer, you must always put the buffer specifier first. For example, if you want to delete lines 4 through 12 in the buffer EXTRA, put the specifiers in this order:

```
*DELETE =EXTRA 4 THRU 12
```

If you are moving the first 10 lines from the current buffer to a certain place in a second buffer, put the second buffer line number after the second buffer name.

```
*MOVE 1 THRU 10 TO =MAIN 23
```

4.4.2 The Range Specifier

The range specifier tells EDT which part of a buffer you want the command to work on. The range specifier can reference lines in the following ways:

- Line numbers
- Position of the text in the buffer
- Strings contained in the lines
- The position of the lines relative to the current line

You can specify a single line, a group of contiguous lines, noncontiguous lines, or entire buffers. Table 4-1 summarizes the different types of line referencing. Table 4-2 lists the symbols and words that you can use with different range types.

Table 4-1: Range Types

Range Type	Description
. (period)	The current line
number	The EDT line number
"string" 'string'	The next line containing the quoted string
BEGIN	The first line of the buffer
END	The last line of the buffer
LAST	The last line EDT was at in the previous buffer
WHOLE	The entire buffer
BEFORE	All lines in the buffer before the current line
REST	All lines in the buffer starting with the current line and ending with the last line
SELECT	A portion of text marked by the keypad SELECT function or the nokeypad SEL command

Table 4-2: Range Specifier Symbols and Words

Symbol/Word	Description
, AND	Used to join noncontiguous ranges in a list. Only single lines can be joined in this way.
: THRU	Indicates a group of lines starting with the first range specifier and ending with the second.
n	Indicates the number of lines away from the current line.
# n FOR n	Indicates that the command is to act on the next n lines.
+	Indicates that string or n refers to a line or lines after the current line.
–	Indicates that string or n refers to a line or lines before the current line.
ALL	Indicates that the command applies to all lines containing the string.

4.4.2.1 Period — When the period (.) or dot is used as a range specifier, it always refers to the current line or to the most recent current line in another buffer. Sometimes EDT assumes you mean the current line when you do not specify a range. However, there are many times when the period is useful. Some of the more specialized uses are described with the line mode commands they apply to, such as SUBSTITUTE and TYPE.

When you are doing a lot of inserting and deleting, the EDT line numbers can become rather long. It is much easier to type a period when you want EDT to print the current line, instead of 485.688, for instance. These sample commands use the period range specifier.

```
*TYPE .
```
Prints the current line.

```
*TYPE . FOR 10
```
Prints the current line and the next nine lines.

```
*DELETE -3 THRU
```
Deletes the current line and the three lines preceding it. The **3** with the minus sign refers to the third line before the current line.

The **n** values (such as 10 and 3 in the previous examples) always indicate a line relative to the current line or a specific line included in the range. For example, 150 +5 means the fifth line after line number 150.

4.4.2.2 Number — Number refers to the EDT line number. This is the number that EDT uses to uniquely identify each line in a buffer. You can always use these numbers to tell EDT precisely which line or lines you want to access. These sample command lines use the EDT line numbers as the range specifier.

```
*TYPE 35
```
Prints line 35 at the terminal.

```
*TYPE 137.3 THRU 142.32
```
Prints all the lines in the group starting with line 137.5 and ending with line 142.32. There is no way to tell from this command how many lines EDT will display.

```
*TYPE . THRU 150
```
Prints all the lines in the group starting with the current line and stopping with line number 150. There is no way to tell from this command how many lines EDT will display.

4.4.2.3 Strings — Strings can reference lines in line mode. When a string is used as a range specifier, it must be enclosed in quotation marks, either single (') or double ("). Line mode generally uses the string to reference the entire line; it does not act on the string itself. Thus, if you use the command DELETE "string", EDT deletes the entire line containing that string.

A minus sign before the quoted string tells EDT to look back toward the beginning of the buffer to find the line containing the string. When ALL precedes a quoted string, it means that every

line in the buffer containing the string is to receive the action of the command. These sample command lines show the string range specifier.

```
*TYPE "December"
```
Displays the first line it encounters containing **December**.

```
*TYPE -"X-RAY"
```
Works backward from the current line to find the line with **X-RAY**. Then displays that line.

```
*TYPE "Dear"+2 THRU "Sincerely"-2
```
Displays the body of a letter, starting with the first paragraph and ending with the last.

```
*TYPE 25 THRU "Portland"
```
Displays the group of lines starting with line number 25 and ending with the line containing **Portland**.

```
*TYPE . THRU 1000 ALL "Date:"
```
Displays every line containing the string **Date:** that appears in the group of lines starting with the current line and ending with line number 1000.

4.4.2.4 BEGIN – BEGIN indicates the first line of a buffer. You can use it as a single line reference or in combination with other range types. These sample command lines use the BEGIN range specifier.

```
*TYPE BEGIN
```
Displays the first line of the current buffer.

```
*TYPE =EXTRA BEGIN THRU "Dear Sir:"
```
Moves to the buffer EXTRA and displays the group of lines starting with the first line in the buffer and ending with the line containing the string **Dear Sir:**.

```
*TYPE BEGIN +6
```
Displays the seventh line of the current buffer.

4.4.2.5 END – END indicates the end of a buffer. You can use it as a single line reference or in combination with other range types. These sample command lines use the END range specifier.

```
*TYPE END
```
Displays the end of buffer mark ([EOB]).

```
*TYPE "January" THRU END
```
Displays the group of lines starting with the line containing the string **January** and ending with the [EOB] mark.

```
*TYPE =SUBROUTINE 185 THRU END
```
Moves to the buffer SUBROUTINE and displays the group of lines starting with 185 and ending with the [EOB] mark.

4.4.2.6 LAST — When you use LAST as the range specifier, EDT moves to the most recent previous buffer. The command that includes the LAST specifier operates on the last current line in that previous buffer. Suppose you move some lines from the MAIN buffer to the PASTE buffer and then make some changes to the lines in the PASTE buffer. To resume editing in the MAIN buffer, you can give the command:

```
*TYPE LAST
```

LAST can only be used by itself. EDT does not accept LAST with other range specifiers or symbols or with buffer names.

4.4.2.7 WHOLE — Use WHOLE when you want EDT to perform your command on the entire buffer. WHOLE can refer either to the current buffer or to a specified buffer. These sample command lines use the WHOLE range specifier.

```
*TYPE WHOLE
```
Displays every line in the current buffer.

```
*MOVE WHOLE TO =MAIN END
```
Moves every line in the current buffer to the end of the MAIN buffer. MAIN becomes the current buffer.

4.4.2.8 BEFORE — BEFORE refers to the group of lines starting with the first line in the current buffer and ending with the line above the current line. BEFORE should not be used in combination with other range specifiers or with a buffer specifier. These sample command lines use the BEFORE range specifier.

```
*TYPE BEFORE
```
Displays the group of lines in the current buffer starting with the first line and ending with the line just before the current line.

```
*DELETE BEFORE
```
Deletes all lines in the current buffer starting with the first line and ending with the line just before the current line.

4.4.2.9 REST — REST refers to the remaining lines in the buffer; that is, all the lines from the current line to the end of the buffer. You should not use REST in combination with other range specifiers or with a buffer specifier. These sample command lines use the REST range specifier.

```
*TYPE REST
```
Displays the group of lines in the current buffer starting with the current line and ending with the end of buffer mark ([EOB]).

```
*MOVE REST TO =ENDING
```
Moves the group of lines in the current buffer starting with the current line and ending with the last line to the buffer named ENDING. ENDING becomes the current buffer.

4.4.2.10 SELECT – SELECT refers to lines that have been marked by a screen mode select range. When you use a select range with a line mode command, you must have only complete lines in the range. Otherwise EDT prints the message **Select complete lines only**. Once you have established the select range, use either the keypad COMMAND function or the nokeypad EXT command to enter the appropriate line mode command.

```
Command:  COPY SELECT TO =ADDRESS1
```

Puts a copy of the select range in the buffer ADDRESS1. Moves to the ADDRESS1 buffer.

```
EXT WRITE SUBPROG.BAS SELECT
```

Puts a copy of the select range in the external file SUBPROG.BAS in the current directory.

4.4.2.11 Specifying Line Numbers That Do Not Exist – When you are using a line number as the range specifier, it is possible for you to specify a number that does not exist in the text you are editing. Whenever this happens, EDT assumes you mean the next highest number it can find. For example, suppose you have moved line 397 to a different part of the buffer causing EDT to give that line a new number. If you specify 397 in a TYPE command later on during your editing session, EDT displays the next line, which might have 397.1, 398, or 397.01 as its line number.

```
*TYPE 397
  397.1       for all subsequent meetings
```

4.4.2.12 Using Range Specifiers with the <null> Command – The <null> command is similar to the line mode TYPE command, but it uses no command word, only a buffer or range specifier. The command is shown in angle brackets to indicate that you do not type the word **null** (or the angle brackets) when using this command. Because there is no command word, EDT requires that you precede some range specifiers with with a percent sign (%). You must use the percent sign whenever the first element of the <null> command statement begins with a letter. For example:

```
*%BEGIN THRU 27
```

Displays the group of lines starting with the first line of the buffer and ending with line 27.

```
*%REST
```

Displays the group of lines starting with the current line and ending with end of buffer mark ([EOB]).

When the specifier word is not the first element in the command line, EDT cannot confuse it with a command word and therefore no percent sign is required.

```
*. THRU END
```

Displays the group of lines starting with the current line and ending with the end of buffer mark. Notice that this command produces the identical results as %REST.

4.4.2.13 Error Messages with Line Ranges — The variety of line range references that you can give with line mode commands is extensive. However, you can make a mistake occasionally. When you type a line range that EDT cannot process, the editor prints an error message and does not process the command at all. Such a message is merely a warning. Try to figure out where the error occurred and type the command again.

4.5 Editing a File with Line Mode

The next sections cover all the steps necessary to edit a file in line mode. First there are explanations of the EXIT, QUIT, and HELP commands that you will use regardless of the editing mode you work in. Next you find out how to insert and delete text so that you can use line mode to create a new file and put something into it. Then you learn how to use the FIND, <null>, and TYPE commands to move around in your editing session and display the various lines of text. By this time, you are ready for the SUBSTITUTE and SUBSTITUTE NEXT commands, which are the workhorses of line editing sessions.

Once you have mastered these commands, you can learn how to manipulate large blocks of text with the COPY, MOVE, and REPLACE commands.

4.6 The Commands That End Your EDT Session — EXIT, QUIT

The EXIT and QUIT commands were introduced in Chapter 2. This section covers these commands in more detail.

When you want to save a copy of the text in the MAIN buffer, use the EXIT command. If you do not want to save a copy of your session, use the QUIT command. The full syntax for each command is:

```
EXIT [file-specification] [/SEQUENCE[:initial[:increment]] ] [/SAVE)

QUIT [/SAVE]
```

EDT allows you to supply a file specification with the EXIT command if you want to copy the MAIN buffer text to an output file with a different specification from the one you used when you started your session. For example, if you are using EDT to modify a letter to Mr. Smith so you can send it to Ms. Jones, you call up the Smith letter file but then specify that the edited version be copied into the Jones letter file.

```
☞ EDIT/EDT SMITHLET.RNO
```

After you have made the necessary changes, use this EXIT command:

```
*EXIT JONESLET.RNO
```

Both the file SMITHLET.RNO and the file JONESLET.RNO now exist in your directory.

When you use the EXIT command, EDT creates a new file. If EDT is editing an existing file, the new version is placed in the same directory as the old file unless you specify otherwise. If you are creating a new file with EDT, the new file is created in the current directory unless you specify otherwise.

EDT can create files with the EXIT command, but it cannot create directories. If you include directory information in your EXIT file specification, that directory must exist somewhere and you must have access to it.

Both EXIT and QUIT take the /SAVE qualifier, which tells EDT to save a copy of the journal file in the current directory. See Chapters 2 and 7 for information on EDT's journal facility. The EXIT command can also take the /SEQUENCE qualifier. Details on using this qualifier with EXIT appear in the section on Using EDT Line Numbers in Chapter 7.

Here are some sample EXIT and QUIT commands.

```
*EXIT
```
Ends your EDT session. Puts a copy of the MAIN buffer text in a file with the same specification that you gave in your EDIT/EDT command line.

```
*EXIT /SAVE
```
Ends your EDT session. Puts a copy of the MAIN buffer text in a file with the same specification that you gave in your EDIT/EDT command line. Saves a copy of the journal file in the current directory.

```
*EXIT NEWLIST.DAT
```
Ends your EDT session. Creates the file NEWLIST.DAT and puts a copy of the MAIN buffer text into it.

```
*EXIT NEWLIST.DAT /SAVE
```
Ends your EDT session. Creates the file NEWLIST.DAT and puts a copy of the MAIN buffer text into it. Saves a copy of the journal file in the current directory.

```
*EXIT /SEQUENCE:100:10
```
Ends your EDT session. Puts a copy of the MAIN buffer text in a file with the same specification that you gave in your EDIT/EDT command line. Adds sequence numbers to the file starting with number 100 and incrementing by 10.

```
*QUIT
```
Ends your EDT session without creating any new file. Any changes you made during your editing session are not recorded.

```
*QUIT /SAVE
```
Ends your EDT session without creating any new file. Saves a copy of the journal file in the current directory. You can use the journal file to recreate the editing session.

4.7 The HELP Facility — HELP

EDT's HELP facility has two parts – one is accessible from line mode; the other from keypad mode. The keypad HELP function is described in Chapter 3. HELP for nokeypad commands is accessed through the line mode part of the HELP facility. Chapter 5 explains how to get HELP for nokeypad commands.

You can use the line mode HELP command any time you see the asterisk prompt (*). The syntax for the HELP command is:

```
HELP [topic [subtopic ...]]
```

Both the topic and subtopic specifiers are optional. If you supply no topic with your HELP command, EDT prints a brief explanation of how to use the HELP facility and a list of the HELP topics. Once you have found the topic you want, type HELP followed by that topic. For example:

```
*HELP EXIT
```

Read the description of the EXIT command. At the end, EDT tells you that additional information is available on two subtopics: /SAVE and /SEQUENCE.

Now look back at the HELP command syntax. Notice that the square brackets surrounding the subtopic are nested within the topic brackets. If you want information on the /SAVE qualifier, type:

```
*HELP EXIT /SAVE
```

If you type HELP /SAVE, EDT prints this message instead of the information you wanted.

```
*HELP /SAVE
Sorry, no documentation on /SAVE
```

Then EDT lists the topics that you can get information on. If you omit the slash before the qualifier subtopic, for instance typing **HELP EXIT SAVE**, EDT tells you that there is no documentation on EXIT SAVE, and this time lists the valid subtopics for the EXIT command.

EDT can help you get information on commands even when you supply only the first letter or so. If you ask for HELP D, EDT prints the information for both the DEFINE and DELETE commands. HELP CH gives you the CHANGE command, but HELP C supplies information on CHANGE, CLEAR, and COPY.

On VT100-type terminals you can use the wildcard character (*) with the HELP command to get information on all subtopics for a command without having to supply the subtopic names. This is helpful when you cannot remember what the subtopics are but you do not need to see the information on the command itself. (This feature is not available on VT52 terminals.) For example, the following command displays the information on TYPE /BRIEF and TYPE /STAY, but not on TYPE itself:

```
*HELP TYPE *
```

Some computer sites have more than one EDT HELP file. The SHOW HELP command tells you which HELP file you are currently using. SET HELP enables you to access a different HELP file if more than one is available. In most instances, you need not concern yourself with different HELP files, but you should be aware that they can exist in case someone at another terminal gets different text from another HELP file.

It is possible to create your own HELP file or modify the HELP file text. Information on this EDT feature appears in Chapter 7.

4.8 Inserting and Deleting Lines — INSERT, DELETE

If you are starting to learn line mode and have no files in your directory, you need to know how to create a file and put some text into it. If you have a file in your directory and want to start by making changes in it, you can skip this section for the time being and go on to the next one to find out how to move from place to place in line mode.

When you call up EDT to edit a file that does not exist, the first thing you want to do is type some text. You need the INSERT command to do that. There are two ways to use the INSERT command. One way handles only a single line of text at a time. If you have nothing to edit, you will not find the one-line INSERT command very useful, so it is discussed later in this chapter.

The basic form of the INSERT command has this syntax:

```
INSERT [range]
```

If you are working with an empty buffer, there will not be any range to specify. All you need to type is INSERT; EDT immediately enters its insert state. You can see that on your paper or screen because the print head or cursor moves over to the right.

Anything you type after the INSERT command is considered text. You can continue to type for as long as you want. Each time you want to start a new line, press the RETURN key to move to the beginning of the next line. When you are finished typing the text, press CTRL/Z to signal EDT that you want to exit from the insert state.

```
☞ EDIT/EDT NEWFILE.DAT

Input file does not exist
[EOB]
*INSERT
              This is a test.  I am creating a file by using EDT.
              Right now I am typing text into the MAIN buffer.
              At the moment there is no file named NEWFILE.DAT in
              my directory.   But when I finish my editing session
              and type the EXIT command, EDT will save a copy of
              this text in my directory under the name NEWFILE.DAT.
              ^Z
[EOB]
*EXIT
DISK$USER:[THOMAS]NEWFILE.DAT;1 6 lines
```

Notice the symbol ^Z at the end of the text. This is what EDT displays when you press CTRL/Z. To issue the CTRL/Z signal to EDT, press the CTRL (control) key first, and while holding it down, press the letter Z key. Do not type the circumflex (^) and then the letter Z.

Typing the EXIT command causes EDT to create the file NEWFILE.DAT. The file does not exist in your directory until EDT has finished processing the EXIT command.

If you print the file at your terminal, it looks exactly the way you typed it. To make any corrections or changes, you need to call up the file so that you can edit it with EDT.

```
☞ EDIT/EDT NEWFILE.DAT

    1       This is a test.  I am using EDT to create a file.
```

This time, EDT does not tell you that the input file does not exist. Instead, it displays the first line of the file, which is the first line in the MAIN buffer. Notice the 1 at the left. This is the number EDT has assigned to the line.

Since you have already created a file and have used EDT successfully at least once, the first line in the sample text is no longer applicable. You can delete it.

The basic syntax for the DELETE command is:

```
DELETE [range]
```

Your text has lines in it and these lines have numbers. Therefore, you can use a range specifier with the DELETE command. Because line 1 is the current line, it is not necessary to give the range specifier. If you type DELETE without the line 1 range specifier, the results are the same.

```
*DELETE 1
1 line deleted
    2       Right now I am typing text into the MAIN buffer.
```

EDT lets you know that one line has been deleted as a result of your command and displays the next line in the buffer.

By this time you might not remember what text is in the buffer. But you do know that there are only a few lines, so you can ask EDT to display all the lines at your terminal.

```
*TYPE WHOLE
    2       Right now I am typing text into the MAIN buffer.
    3       At the moment there is no file named NEWFILE.DAT in
    4       my directory.   But when I finish my editing session
    5       and type the EXIT command, EDT will save a copy of
    6       this text in my directory under the name NEWFILE.DAT.
[EOB]
```

Line 1 is gone; the first line is now number 2. After you read the text, you see that lines 3 through the end are no longer applicable either. To delete those lines, type:

```
*DELETE 3 THRU END
4 lines deleted
[EOB]
```

EDT is now positioned at the end of the buffer, after line 2, which is the only line in your buffer at the moment.

To add more text, use the INSERT command again. If you are not absolutely sure which line is current, you can use the END range specifier with your INSERT command to ensure that the new text will be inserted at the end of the buffer.

```
*INSERT END
            I have already created a file called NEWFILE.DAT.
            When I use EXIT to end this session, I will have
            two versions of NEWFILE.DAT in my directory.
            Because I have deleted lines and am adding new
            text, the two versions will be different.
```

```
This is a new paragraph.  I pressed the RETURN key
twice in succession to create the blank line
that separates the two paragraphs.  Although a blank
line does not have any text in it, EDT assigns it a
a number, nevertheless.
^Z

[EOB]
```

Suppose you decide to eliminate the first line of the text you inserted. It is the second line in the buffer, but if you ask EDT to DELETE 2, the line **Right now I am ...** is deleted. EDT does not renumber the entire buffer when you insert text. Because you deleted all lines after number 2, EDT was able to number the inserted text starting with 3 for the first line of the added text, and then assign whole numbers in ascending order. Thus, to delete the first line of the new text you can use this command:

```
*DELETE 3
1 line deleted
    4       When I use EXIT to end this session, I will have
```

There are times when the EDT line numbers become long and complex. In these instances, you can use other types of line references to tell EDT which lines to operate on. Suppose you start with this text in line 1 and insert five lines above it.

```
    1       This is line number 1.

*INSERT
            I am typing five lines above line 1.  The lines
            will have line numbers with decimal fractions.
            This is the line I plan to delete.
            I will show four different ways to
            reference the line I want deleted.
            ^Z

    1       This is line number 1.
```

Notice that line 1 is still the current line. The new text appears above line 1. Using the TYPE command, you can have EDT display the first six lines in the buffer.

```
*TYPE 0 THRU 1
    0.1     I am typing five lines above line 1.  The lines
    0.2     will have line numbers with decimal fractions.
    0.3     This is the line I plan to delete.
    0.4     I will show four different ways to
    0.5     reference the line I want deleted.
    1       This is line number 1.
```

After EDT processes this TYPE command, line 0.1 becomes the current line. Here are four ways to reference line 0.3:

```
1.      *DELETE 0.3

2.      *DELETE "plan"
```

```
3.      *DELETE BEGIN + 2

4.      *DELETE 1 -3
```

The first command simply uses the line number itself. The second command relies on your knowing that the string **plan** appears in that line. The third command assumes that you know that the line is the third line in the buffer. The fourth command assumes that you know the line is three lines before line number 1.

4.9 Moving from Place to Place — FIND, <null>, TYPE

In addition to the TYPE command, which you have already seen in some examples, EDT has two other commands that move you from place to place during your editing session: <null> and FIND. All three commands perform similar functions but with minor differences in syntax or effect. The command syntaxes are:

```
FIND [=buffer] [range]

[=buffer] [[%]range]

TYPE [=buffer] [range] [/BRIEF[:n]] [/STAY]
```

The <null> command is an implied TYPE command. Both <null> and TYPE have the same function, but use slightly different syntax. The <null> command consists simply of a buffer and/or range specifier with no command word. To avoid confusing EDT, you must use a percent sign (%) before range specifiers that are words, for example, %BEGIN or %END, whenever the word is the first element in the <null> command. However, if your <null> command is **. THRU END**, you do not need the percent sign before the word END. The TYPE command takes two qualifiers; <null> has none.

The major difference between the FIND and TYPE commands is that FIND only moves EDT to the new position; FIND never causes EDT to display any text at the terminal. Thus there is no point in using more than a single line as the range specifier with FIND.

The TYPE command displays all the lines specified by the range. You can use TYPE to have EDT display lines 10 through 20, or display lines 3, 48, and 58. With TYPE, you frequently use a multiline range specifier.

Whenever you specify more than one line for the range with TYPE or <null>, EDT displays each line in the range and then positions itself at the first line of the range. Suppose you give either of these commands:

```
*TYPE 25 THRU 128

*TYPE 25, 36, 47, 68, 89, 93, 101, 113, 128
```

As soon as EDT displays all the lines in the range, it positions itself at line 25, the first line in both ranges. Line 25 becomes the current line.

If you give only a buffer specifier with the TYPE command, EDT moves to that buffer, displays every line in the buffer and leaves you at the end. If you give only a buffer specifier with the FIND command, EDT positions itself at the first line of that buffer.

This example uses text that makes it easy for you to remember the number of each line. Enter the text using the INSERT command; then use TYPE WHOLE to have EDT display the lines with their line numbers.

```
☞ EDIT/EDT TEST.DAT
Input file does not exist
[EOB]
*INSERT
            One partridge in a pear tree
            Two turtle doves
            Three French hens
            Four calling birds
            Five gold rings
            Six geese a-laying
            Seven swans a-swimming
            Eight maids a-milking
            Nine ladies dancing
            Ten lords a-leaping
            Eleven pipers piping
            Twelve drummers drumming
            ^Z
[EOB]

*TYPE WHOLE
    1       One partridge in a pear tree
    2       Two turtle doves
    3       Three French hens
      .
      .
      .
   11       Eleven pipers piping
   12       Twelve drummers drumming
[EOB]
*
```

The following examples show various uses of FIND, <null>, and TYPE with the sample text:

```
*TYPE 9
    9       Nine ladies dancing

*%END
[EOB]

*FIND BEGIN                    (Remember that FIND displays nothing.)

*+1
    2       Two turtle doves

*TYPE 5 THRU 7
    5       Five gold rings
    6       Six geese a-laying
    7       Seven swans a-swimming

*"three"
String was not found

*-"three"
    3       Three French hens
```

```
*FIND "four" +6

*TYPE .
   10        Ten lords a-leaping

*%REST
   10        Ten lords a-leaping
   11        Eleven pipers piping
   12        Twelve drummers drumming
[EOB]

*FIND -"five"

*%BEFORE
    1        One partridge in a pear tree
    2        Two turtle doves
    3        Three French hens
    4        Four calling birds

*TYPE 8-1
    7        Seven swans a-swimming

*. +2
    9        Nine ladies dancing

*TYPE 2, 4, 6
    2        Two turtle doves
    4        Four calling birds
    6        Six geese a-laying

*%ALL 'en'
    3        Three French hens
    7        Seven swans a-swimming
   10        Ten lords a-leaping
   11        Eleven pipers piping
```

As you can see, there are many ways to use the range specifier.

4.10 Performing Substitutions — SUBSTITUTE, SUBSTITUTE NEXT

So far you have seen how to insert and delete lines and how to move around in the MAIN buffer. The next commands to learn are the SUBSTITUTE commands. These enable you to make changes to your text without having to delete and insert entire lines.

There are two commands that perform substitutions in line mode: SUBSTITUTE and SUBSTITUTE NEXT. (The latter is often called the NEXT command because the word SUBSTITUTE is optional in the syntax.) Both perform the same basic function of substituting one string for another. The SUBSTITUTE command can take several qualifiers – /BRIEF, /QUERY, and /NOTYPE. SUBSTITUTE NEXT accepts no qualifiers. SUBSTITUTE requires that you supply a range specifier if the string you want to replace is not on the current line. SUBSTITUTE NEXT looks for the next occurrence of the string you want to change and therefore needs no range specifier.

Here is the syntax for both commands:

```
SUBSTITUTE/[string-1]/string-2/ [=buffer] [range]
          [/BRIEF[:n]] [/QUERY] [/NOTYPE]

[SUBSTITUTE] NEXT[/[string-1]/string-2/]
```

4.10.1 SUBSTITUTE

The SUBSTITUTE command looks complex with its three optional qualifiers. Although qualifiers are not discussed until the end of this chapter, you should be aware that the slashes (/) surrounding string-1 and string-2 are delimiters; the slashes preceding the qualifier words are signals to EDT that the following letter or word is a qualifier. You can replace the delimiter slashes with any nonalphanumeric character except the percent sign (%) and the underscore (_), for example |, (, or '. However, the delimiter character that you choose must not appear either in string-1 or in string-2. Note that the qualifier signal must always be the slash.

```
SUBSTITUTE/[string-1]/string-2/ [=buffer] [range]
          [/BRIEF[:n]] [/QUERY] [/NOTYPE]
```

String-1 and string-2 have the same references in both commands. String-1 is always the string you want to change; string-2 is always the replacement string. String-1 is generally referred to as the search string because you want EDT to find it and then delete it. String-2 is called the substitute string because it is substituted for string-1. Although string-1 and string-2 are both optional, you must supply at least one of them with your command. Notice that the delimiters are *not* optional.

In Chapter 2, you read that there were some buffers that you could not enter or edit, and that did not appear in the SHOW BUFFER list. Two of these are the search buffer and the substitute buffer. Whenever you use a substitute command, EDT stores string-1 (the search string) in the search buffer and string-2 (the substitute string) in the substitute buffer.

The search buffer contents are also affected by operations that locate strings such as FIND, <null>, and TYPE in line mode; FIND in keypad mode; and "move", S, and SN in nokeypad mode. The line mode string range specifier affects the search buffer contents. (The nokeypad string entity also affects the contents of the search buffer.) Each time you use any of these commands, functions, or specifiers, EDT replaces the contents of the search buffer with the new search string. So you must be aware of the current search string when you omit string-1 from a SUBSTITUTE command.

A string can be one or more contiguous characters. When choosing the search string for your SUBSTITUTE command, make sure that EDT will not confuse the string with something else and replace the wrong text.

You must also be aware of the way EDT makes searches. Normally, EDT disregards both case of letters and diacritical marks when making searches. Thus, **SPAT, Spat, spat, Spät,** and **spät** are all the same to EDT. (For more information on how EDT handles search strings, see the section on SET SEARCH in Chapter 6.)

Although EDT normally ignores case differences and diacritical marks in the search string, you must type the substitute string (string-2) exactly as you want it to appear in the text. For example, if you want to change Harry to Mark, you can type Harry (string-1) as /HARRY/, /Harry/, or /harry/, but you must type Mark (string-2) as /Mark/ if you want it to appear that way.

The effects of the SUBSTITUTE command are limited if you supply no range specifier. All the command can do is make one substitution in the current line. If string-1 is not on the current line, no substitutions are made.

```
    35        in teh morning.  Then after that session is over,

*SUBSTITUTE/teh/the/
    35        in the morning.  Then after that session is over,
1 substitution

*SUBSTITUTE/1:00/1:30/
No substitutions
```

If you have more than one substitution to make on the current line, you must supply a range specifier to have all the substitutions made with a single command. If you do not, EDT performs the substitution only on the first string-1 in the line.

```
    45        January 14, 1983.  Then on January 27, 1983, we

*SUBSTITUTE/83/84/
    45        January 14, 1984.  Then on January 27, 1983, we
1 substitution
```

In order to have EDT make both substitutions, supply either the line number (45) or the period (.) for range:

```
    45        January 14, 1983.  Then on January 27, 1983, we

*SUBSTITUTE/83/84/ .
    45        January 14, 1984.  Then on January 27, 1984, we
2 substitutions
```

In addition to letting you know how many substitutions were made, EDT displays the affected line or lines so that you can see the changes.

You can use other range specifiers with the SUBSTITUTE command to perform different numbers of substitutions. With the WHOLE specifier, you can replace every occurrence of string-1 in the buffer with string-2. If you know that you need to replace all occurrences of a string in the next 50 lines or so, you can use a range specifier to have EDT do that. Using the buffer specifier, you can make substitutions in another buffer. Remember if you include a buffer specifier with the SUBSTITUTE command, EDT moves to the specified buffer, performs the substitutions, and remains in that buffer. If you use a buffer specifier with no range, EDT performs the substitution throughout the named buffer.

When you use a range specifier with SUBSTITUTE, EDT stops on the first referenced line. If a group of lines is referenced (for example, 1 THRU 10 or "January" THRU +40), EDT returns to the first line of the range after performing all the substitutions. Even if no substitutions were made on the first line of the range, EDT still moves to that line.

```
*SUBSTITUTE/Orleens/Orleans/ 34 THRU 84
    45        at the winter meetings to be held in New Orleans
    53        Make your reservations early, as New Orleans
    59        New Orleans has many excellent French restaurants.
    65        famous for its New Orleans jazz bands.  Also,
    83        to coincide with New Orleans' Mardi Gras festival.
5 substitutions
```

```
*TYPE .
   34        This year we are planning a special conference.

*SUBSTITUTE/December 83/January 84/ 10 THRU 53
No substitutions

*TYPE .
   10        but have not yet signed the contract.  When they
```

4.10.2 SUBSTITUTE NEXT

The SUBSTITUTE NEXT command can take the string-1 and string-2 specifiers, but cannot take a range or buffer specifier. SUBSTITUTE NEXT uses string-1 as the search string and moves to the next occurrence in the current buffer. If EDT finds string-1, it replaces that string with string-2. Only one substitution can be performed by each SUBSTITUTE NEXT command. Therefore, EDT does not bother to display the number ot substitutions made. SUBSTITUTE NEXT has no qualifiers.

```
[SUBSTITUTE] NEXT[/[string-1]/string-2/]
```

EDT does display the line on which the substitution took place so that you can verify the change. If you want to repeat the substitution for the next string-1 in the buffer, simply give the SUBSTITUTE NEXT command with no strings. This version of the command tells EDT to use the current contents of the search and substitute buffers.

Since SUBSTITUTE NEXT does not take a buffer specifier, it can only be used in the current buffer. Also, it can only move forward, starting its search at the current line and continuing until it either finds string-1 or reaches the end of the current buffer. Because it takes no range specifier, the SUBSTITUTE NEXT command leaves EDT at the line in which the substitution takes place. If string-1 cannot be found, SUBSTITUTE NEXT leaves EDT at the end of the buffer; no line is printed, just the asterisk prompt (*) indicating that EDT is ready for a new command. You must move back toward the beginning of the buffer before you can issue another SUBSTITUTE NEXT command.

```
*TYPE .
    5        Fruits are a healthy source of vitamin C.  You

*SUBSTITUTE NEXT/apple/orange/
    8        An orange contains more vitamin C than a peach.

*SUBSTITUTE NEXT/mango /banana/
*TYPE .
[EOB]
```

Notice that when you try to substitute **banana** for **mango**, EDT cannot find **mango** anywhere from line 8 to the end of the buffer. Therefore, it moves to the end of the buffer and stops at the [EOB] mark. The TYPE command verifies that your current line is the end of buffer mark.

If you are sure that **mango** is somewhere in the buffer, move to the first line and ask EDT to make the substitution again. This time, you only need to type the SUBSTITUTE NEXT command itself, because string-1 and string-2 are already in the search and substitute buffers.

```
*FIND 1

*SUBSTITUTE NEXT
    7        Some tropical fruits, such as pineapples, bananas,
```

The syntax for SUBSTITUTE NEXT indicates that the command word SUBSTITUTE is optional or can be abbreviated to S. If you include the word SUBSTITUTE or its abbreviation in your command line, you must use a space to separate it from the word NEXT or its abbreviation N.

```
SUBSTITUTE NEXT

S N
```

4.10.3 Comparing SUBSTITUTE and SUBSTITUTE NEXT

Table 4-3 compares the SUBSTITUTE and SUBSTITUTE NEXT commands with regard to specifiers, qualifiers, number of substitutions, search direction, and EDT display.

Table 4-3: Comparison of SUBSTITUTE and SUBSTITUTE NEXT Commands

	SUBSTITUTE	SUBSTITUTE NEXT
buffer	the specified buffer	only the current buffer
range	the specified range	only the next occurrence of string-1
number of substitutions possible	as many as there are search strings in the range or buffer	only 1
qualifiers	/BRIEF, /QUERY, /NOTYPE	none
changed line(s) displayed	yes	yes
number of substitutions displayed	yes	no
current line: successful search	first line of range	line on which substitution was made
current line: unsuccessful search	first line of range	end of buffer ([EOB])
search direction	depends on range	forward only
strings	must include one string	both are optional

You do not have to include either the search string or the substitute string with SUBSTITUTE NEXT. However, for SUBSTITUTE, you must include at least one of the strings. The following examples summarize the possibilities for each command:

```
SUBSTITUTE:

This is file A.      SUBSTITUTE/file/buffer/          This is buffer A.
This is file B.      SUBSTITUTE///          Search string may not be null
This is file C.      SUBSTITUTE/file//                This is  C.
This is file D.      SUBSTITUTE//buffer/              This is buffer D.
This is file E.      SUBSTITUTE//buffer/ "e."        *This is file buffer
```

*In this line, EDT first located the appropriate line using the search string **e.** to find it. The search buffer no longer contains the string **file**; the current search string is **e.** So, EDT replaces

the string **e.** with the substitute string. Only use a string range specifier if you include string-1 in your SUBSTITUTE command or if you want to replace the string range specifier with the substitute string.

```
SUBSTITUTE NEXT:

This is file A.     SUBSTITUTE NEXT/file/buffer/          This is buffer A.
This is file B.     SUBSTITUTE NEXT///       Search string may not be null
This is file C.     SUBSTITUTE NEXT/file//                This is  C.
This is file D.     SUBSTITUTE NEXT//buffer/              This is buffer D.
This is file E.     SUBSTITUTE NEXT                       This is buffer E.
```

4.11 Line Editing in Action — Sample Session

You now know enough line mode commands to do almost any type of line mode editing. The TYPE, <null>, and FIND commands move EDT from place to place. INSERT and DELETE add and remove lines of text. SUBSTITUTE and SUBSTITUTE NEXT allow you to make small changes in any text.

The next example shows how to enter a memo and then revise it later on. The editorial changes have been made on the version as it was first entered. You need only seven line mode commands to edit the entire file.

The first thing you see is the sample session that puts the memo text into the MAIN buffer and then creates the file containing the text when EXIT is typed.

```
☞ EDIT/EDT SECRET.MMO
Input file does not exist
[EOB]
*INSERT
                DATE:       April 14, 1984

                TO:         All Sales Representatives

                FROM:       George B. Shaver

                SUBJECT:    New Products

                It has come to my attention that information
                on new products has been leaded to the by
                someone in the company.  While no one has
                pointed a finger at the sales staff, all
                vice presidents have been asked to notify
                the members of their department about the
                importance of secrecy in these matters.

                As you know we are having a major press
                conference next month to make announcements
                of our new product lines.  Even though this
                event is only four weeks away, information
                leaks could give our competition the
                advantage it needs to scoop us.  In a
                business as highly competitive as ours, we
                need every edge we can get.
```

(continued on next page)

```
Since sales reps receive can only stand
to lose valuable commissions from violating
secrecy, I doubt that any of you are
involved in the leaks.  However, severe
measures will be taken with any employee who
decides to resign from our staff and then
takes a job with one of our competitors.  If
anyone has plans to do so, the burden of
proof will be on him to show that such plans
have nothing to do with leaks of
information.

Unfortunately, we can do nothing about the
fact that our new rubber duck model has
already been pre-announced.  Any advance
word on Cuddles the Bear will result in
firings!!

Rewards will be given to those who report any
leaks to me.
^Z
[EOB]
*EXIT
DISK$USER:[SHAVER]SECRET.MMO;1 45
```

Figure 4-1 shows the printed copy of the memo with the editorial changes marked.

Figure 4-1: Sample Memo

```
DATE:       April 11, 1984

TO:         All Sales Representatives

FROM:       George B. Shaver
                ^ Vice President, Sales
SUBJECT:    New Products

It has come to my attention that information
on new products has been leaded to the by
someone in the company.  While no one has
pointed a finger at the sales staff, all
vice presidents have been asked to notify
the members of their department about the
importance of secrecy in these matters.

As you know, we are having a major press
conference next month to make announcements
of our new product lines.  Even though this
event is only four weeks away, information
leaks could give our competition the
advantage it needs to scoop us.  In a
business as highly competitive as ours, we
need every edge we can get.

Since sales reps receive can only stand
to lose valuable commissions from violating
secrecy, I doubt that any of you are
involved in the leaks.  However, severe
```

```
measures will be taken with any employee who
decides to resign from our staff and then
takes a job with one of our competitors.  If
anyone has plans to do so, the burden of
proof will be on him to show that such plans
have nothing to do with leaks of
information.

Unfortunately, we can do nothing about the
fact that our new rubber duck model has
already been pre-announced.  Any advance
word on Cuddles the Bear will result in
firings!!

Rewards will be given to those who report any
leaks to me.
```

Here is how your editing session might look:

```
☞ EDIT/EDT SECRET.MMO
    1       DATE:       April 11, 1984

*TYPE "shaver"
    5       FROM:       George B. Shaver

*INSERT .+1
                        Vice President, Sales
                    ^Z
    6

*TYPE 5 THRU 7
    5       FROM:       George B. Shaver
    5.1                 Vice President, Sales
    6
    7       SUBJECT:    New Products

*"lead"
   10       on new products has been leaded to the by

*SUBSTITUTE/lead/leak/
   10       on new products has been leaked to the by
1 substitution

*SUBSTITUTE/by/press by/
   10       on new products has been leaked to the press by
1 substitution

*SUBSTITUTE NEXT/department/departments/
   14       the members of their departments about the

*SUBSTITUTE/secrecy/confidentiality/ REST
   15       importance of confidentiality in these matters.
   28       confidentiality, I doubt that any of you are
2 substitutions

*TYPE .
   14       the members of their departments about the
```

```
*SUBSTITUTE NEXT/know/know,/
   17        As you know, we are having a major press

*SUBSTITUTE/are having/have set up/
   17        As you know, we have set up a major press
1 substitution

*SUBSTITUTE/make announcements/announce/ +1
   18        conference next month to announce
1 substitution

*SUBSTITUTE NEXT/of our/seven/
   19        seven new product lines.  Even though this

*SUBSTITUTE/^lines/s/
   19        seven new products.  Even though this
1 substitution

*SUBSTITUTE NEXT/four/three/
   20        event is only three weeks away, information

*SUBSTITUTE/us/our news/ "scoop"
   22        advantage it needs to scoop our news.  In a
1 substitution

*SUBSTITUTE/,/^is,/ +1
   23        business as highly competitive as ours is, we
1 substitution

*SUBSTITUTE NEXT/s receive/resentatives/
   26        Since sales representatives can only stand

*SUBSTITUTE/are/is/ "doubt"
   28        confidentiality, I doubt that any of you is
1 substitution

*SUBSTITUTE NEXT*him*him/her*
   34        proof will be on him/her to show that such plans

*SUBSTITUTE/leaks of/the recent information leaks./ +1
   35        have nothing to do with the recent information leaks.
1 substitution

*DELETE +1
1 line deleted
   37

*SUBSTITUTE NEXT/any/But, any/
   40        already been pre-announced.  But, any advance

*SUBSTITUTE/the Bear/Bear/ "bear"
   41        word on Cuddles Bear will result in
1 substitution

*DELETE +1
1 line deleted
   43
```

```
*INSERT
                firings!  The same goes for Chuckles Clown!
                ^Z
   43

*DELETE . THRU END
3 lines deleted
[EOB]

*EXIT
DISK$USER:[SHAVER]SECRET.MMO;2 48 lines
```

The final text looks like this:

Figure 4-2: Corrected Memo

```
DATE:       April 11, 1984

TO:         All Sales Representatives

FROM:       George B. Shaver
            Vice President, Sales

SUBJECT:    New Products

It has come to my attention that information
on new products has been leaked to the press by
someone in the company.  While no one has
pointed a finger at the sales staff, all
vice presidents have been asked to notify
the members of their departments about the
importance of confidentiality in these matters.

As you know, we have set up a major press
conference next month to announce
seven new products.  Even though this
event is only three weeks away, information
leaks could give our competition the
advantage it needs to scoop our news.  In a
business as highly competitive as ours is, we
need every edge we can get.

Since sales representatives can only stand
to lose valuable commissions from violating
confidentiality, I doubt that any of you is
involved in the leaks.  However, severe
measures will be taken with any employee who
decides to resign from our staff and then
takes a job with one of our competitors.  If
anyone has plans to do so, the burden of
proof will be on him/her to show that such plans
have nothing to do with the recent information leaks.

Unfortunately, we can do nothing about the
fact that our new rubber duck model has
already been pre-announced.  But, any advance
word on Cuddles Bear will result in
firings!  The same goes for Chuckles Clown!
```

4.12 Using Different Buffers in Line Mode

You can go through hundreds of EDT editing sessions without using any buffer other than MAIN. But additional buffers can help you with special tasks. For example, you can put the contents of another file into a separate EDT buffer during your editing session and then either read the contents to get information or copy portions of that file into your text so you do not have to retype or recreate them. You can use buffers to store portions of your MAIN buffer text until you need them later on during your session. You can put part of your text in a buffer, edit only that portion and then move the edited version back into the main text. You can store a form letter in a buffer and then copy it into MAIN to personalize it for different recipients.

To indicate a different buffer in a line mode command use either:

1. The equal sign (=) immediately preceding the buffer name (= BUFF1, for example)
2. The word BUFFER followed by the buffer name (BUFFER BUFF1, for example)

Either way is acceptable in those line mode commands that take a buffer name as a specifier, except for the CLEAR command. The CLEAR command takes only the buffer name, without either BUFFER or the equal sign (=).

```
CLEAR EXTRA
DELETE =EXTRA
FIND BUFFER EXTRA
```

The line mode commands which take the buffer specifier are:

CLEAR	FIND	<null>	SUBSTITUTE
COPY	INCLUDE	PRINT	TAB ADJUST
DELETE	INSERT	REPLACE	TYPE
FILL	MOVE	RESEQUENCE	WRITE

Whenever you specify a buffer name with one of these commands (except for CLEAR), EDT creates a buffer with that name if one does not already exist. Many commands in the list cannot perform their functions if there is no text already in the buffer that you name. However, if you accidentally type a nonexistent buffer name, even these commands will create a new buffer. For example, the command DELETE =EXTRA 5 THRU 17 normally moves EDT to the buffer named EXTRA and deletes the specified lines. If your editing session has no buffer named EXTRA, EDT creates one, moves there, and then tells you that no lines were deleted.

Most line mode commands move EDT to the buffer specified in the command line. CLEAR, PRINT, and WRITE do not move EDT to the named buffer. When you use a buffer specifier with WRITE or PRINT, EDT always remains in the current buffer. With CLEAR, EDT remains in the current buffer unless you want to delete the current buffer. In that case, EDT moves to the MAIN buffer. If you use CLEAR to empty the MAIN buffer, EDT stays in MAIN.

The COPY and MOVE commands can take two buffer specifiers. The first one tells EDT where to find the text that you want to move or copy and the second one tells EDT the new location for the text.

4.12.1 SHOW BUFFER

When moving from one buffer to another with a computer editor, it is easy to lose track of which buffer you are in. The SHOW BUFFER command displays the names of all buffers that currently exist for your EDT session, shows which is the current buffer, and displays the number of lines in each buffer. (See Chapter 6 for more details on the SHOW BUFFER command.)

Suppose you have a buffer with 28 lines in it and you want to eliminate 13 of them. You type the command:

```
*DELETE =EXTRA 5 THRU 17
No lines deleted
[EOB]
```

You are surprised to find the current buffer empty and your DELETE command wasted. Now, type the SHOW BUFFER command.

```
*SHOW BUFFER
=EXTRA    No        lines
 ADDITION           28        lines
 MAIN     193       lines
 PASTE    No        lines
```

The equal sign (=) points to the current buffer, which is empty. The ADDITION buffer has 28 lines in it. You wanted to delete lines 5 through 17 in that buffer, but forgot the name you had given it. You can issue your DELETE command from the EXTRA buffer and then return to the MAIN buffer to continue your work.

```
*DELETE =ADDITION 5 THRU 17
13 lines deleted
*FIND =MAIN .
*
```

You can use the FIND command to return to the MAIN buffer. Note the period (.) after the buffer name. This signals EDT to return to the last line on which you were working in the specified buffer.

When you use the TYPE command without a range specifier, EDT displays every line in the buffer. If your buffer is long, you might not want to wait for EDT to print all the lines. You can use CTRL/C to stop the TYPE command. After EDT receives the CTRL/C interrupt signal, it stops printing the text and prints the message:

```
Aborted by CTRL/C
```

You can use CTRL/C to abort other EDT commands, such as <null>. See Chapter 8 for more information on CTRL/C.

4.12.2 FIND, <null>, and TYPE with a Buffer Specifier

Moving from buffer to buffer with the FIND, <null>, and TYPE commands is easy as long as you remember that <null> and TYPE cause EDT to print out the entire contents of the buffer when you do not include a range specifier. Table 4-4 summarizes what happens when you use a buffer specifier with these three commands, first without any range specifier, then with the period range specifier, and finally with any other range specifier.

Table 4-4: Commands that Move EDT's Position in Your Editing Session

Command	Action
*FIND = buffer	Moves to the buffer and prints nothing. The first line of the buffer becomes the current line.
*FIND = buffer .	Moves to the buffer and prints nothing. The the last line EDT worked on in the specified buffer becomes the current line.
*FIND = buffer range	Moves to the buffer and prints nothing. The range is a single line reference, which becomes the current line.
* = buffer	Moves to the buffer and prints the entire contents. The first line of the buffer becomes the current line.
* = buffer .	Moves to the buffer and prints the last line EDT worked on in the specified buffer. That line becomes the current line.
* = buffer range	Moves to the buffer and prints all the lines in the range. The first line of the range becomes the current line.
*TYPE = buffer	Moves to the buffer and prints the entire contents. The first line of the buffer becomes the current line.
*TYPE = buffer .	Moves to the buffer and prints the last line EDT worked on in the specified buffer. That line becomes the current line.
*TYPE = buffer range	Moves to the buffer and prints all the lines in the range. The first line of the range becomes the current line.

4.12.3 Deleting a Buffer with CLEAR

The CLEAR command requires a buffer name as a specifier but does not use either the equal sign or the BUFFER signal. The signal is unnecessary because CLEAR cannot take any other kind of specifier or qualifier. The syntax is:

```
CLEAR buffer
```

When you use the CLEAR command, EDT not only deletes the contents of the buffer, but also the buffer itself. Thus, the next time you type SHOW BUFFER, the name of the deleted buffer no longer appears on the list.

There are two exceptions: MAIN and PASTE. Because EDT automatically creates these buffers at the start of each editing session, it does not allow you to delete them. The CLEAR command only empties these buffers. Their names always appear on the SHOW BUFFER list:

```
*SHOW BUFFER
 SECT1    58       lines
 SECT2    125      lines
 SECT3    37       lines
=MAIN     674      lines
 PASTE    11       lines

*CLEAR SECT3
*CLEAR PASTE

*SHOW BUFFER
 SECT1    58       lines
 SECT2    125      lines
=MAIN     674      lines
 PASTE    No       lines
```

If you delete the buffer you are currently working in, EDT moves you to the most recent current line in the MAIN buffer. When you clear the MAIN buffer, the current line becomes the [EOB] mark because there are no lines left in the buffer.

4.13 Moving Blocks of Text – COPY, MOVE, and REPLACE

The COPY, MOVE, and REPLACE commands enable you to easily copy, delete, insert, and move blocks of text.

The COPY and MOVE commands have similar syntax. The difference is that when you move a range of lines or the contents of a buffer, you delete the lines from their original location and put them in a new location. When you use the COPY command, your original text is not altered in any way. EDT simply makes a copy of the lines and places the copy in the new location; the original lines remain in their original location.

The REPLACE command is a combination of DELETE and INSERT. First EDT deletes the lines or buffer that you specify and then shifts to the insert state. When you finish typing the new text, press CTRL/Z to return to line mode's asterisk prompt.

4.13.1 MOVE

MOVE makes it possible for you to change the location of one or more lines in your editing session. You can use the MOVE command to alphabetize lists, change the order of paragraphs in your text, move text to another buffer, and so forth. The MOVE command deletes the lines from the first specified location and puts a copy of them in the second. EDT renumbers the moved lines so that the numbers fit in the new location. You can include a buffer specifier and a range specifier for both locations.

```
MOVE [=buffer-1] [range-1] TO [=buffer-2] [range-2] [/QUERY]
```

Think of the command syntax as "MOVE the text in location-1 TO location-2". Both locations are optional. If neither is included, the net effect on your text is that EDT moves to the line below its initial position and that line becomes the new current line. (The line number of the line EDT "moved" is likely to change.)

For location-1 or location-2 you can supply either a buffer name or a range or both. EDT assumes the current buffer when you omit the buffer specifier and the current line when you omit the range specifier.

The /QUERY qualifier allows you to decide which lines in a range or buffer you want EDT to move. For details about the /QUERY qualifier, see the section on Line Mode Qualifiers at the end of this chapter.

The syntax of the simplest MOVE operation is:

```
MOVE range-1 TO range-2
```

Range-2 always refers to a single line. EDT inserts the line or lines specified by range-1 just above the range-2 line and renumbers the lines. Here is a list of Australian cities to alphabetize:

```
1       Adelaide
2       Canberra
3       Brisbane
4       Melbourne
```

You need to move line 2 in between lines 3 and 4 so that the list is in the correct order.

```
*MOVE 2 TO 4
1 line moved
*TYPE WHOLE
    1       Adelaide
    3       Brisbane
    3.1     Canberra
    4       Melbourne
[EOB]
```

The next example shows how to move several lines of text with the MOVE command.

```
  1       Mr. John Bartholomew
  2       3857 Hudson Street
  3       Montgomery, OH  45242
  4
  5       Ms. Agnes Johnson
  6       2143 Maple Avenue
  7       Goshen, IN  46526
  8
  9       Mr. Peter Chatterton
 10       475 Market Street
 11       Eureka, IL  61530
 12
```

```
*MOVE 4 THRU 7 TO 12
4 lines moved
*TYPE WHOLE
     1        Mr. John Bartholomew
     2        3857 Hudson Street
     3        Montgomery, OH  45242
     8
     9        Mr. Peter Chatterton
    10        475 Market Street
    11        Eureka, IL  61530
    11.1
    11.2      Ms. Agnes Johnson
    11.3      2143 Maple Avenue
    11.4      Goshen, IN  46526
    12
[EOB]
```

You can use the buffer specifier to move lines from one buffer to another. If you specify buffer-1, it must already exist in your EDT session in order for any text to be moved. If buffer-1 does not exist, EDT creates it, and then prints the message **No lines moved**. If buffer-2 does not exist, EDT creates it before moving the text there.

If you do not specify either buffer-1 or buffer-2 in your MOVE command, the move is made within the current buffer. If you specify only buffer-1, EDT moves the lines in that buffer to the current buffer. Conversely, if you specify only buffer-2, EDT moves the lines in the current buffer to buffer-2. After processing a MOVE command, EDT always leaves you in the original buffer or in buffer-2; EDT never stops in buffer-1.

If you do not use a range specifier with buffer-1, EDT moves the entire contents of the buffer. If you omit the range specifier with buffer-2, EDT inserts the moved text above the first line of the buffer. If you are using the MOVE command to create buffer-2, that buffer will be empty, so there is no need to specify a range.

If you omit both buffer-1 and range-1, be sure that you know what the current line is. If EDT is at the end of the current buffer when you issue a MOVE command without specifying location-1, you get the message **No lines moved**.

4.13.2 COPY

The COPY command saves you from retyping material that already exists. When you use COPY, EDT makes a copy of the lines you want to duplicate and places the copy in the new location. The original text remains in the same spot, unchanged in any way. (To copy material from an external file into your editing session, you must use the INCLUDE command, discussed later in this chapter.)

The COPY command takes two qualifiers: /QUERY and /DUPLICATE. The /QUERY qualifier is discussed in the section on qualifiers, which appears at the end of this chapter. However, since COPY is the only command that uses the /DUPLICATE qualifier, you will learn about that qualifier in this section.

The complete syntax of the COPY command is:

```
COPY [=buffer-1] [range-1] TO [=buffer-2] [range-2] [/QUERY] [/DUPLICATE:n]
```

Think of the command syntax as "COPY the text in location-1 TO location-2". Both locations are optional. If neither is included, EDT makes a copy of the current line and puts it above the current line. EDT always assumes the current line in the current buffer when you do not supply a location.

Range-2 is always a single line reference; range-1 can be any number of lines. The simplest form of the COPY command is:

```
COPY range-1 TO range-2
```

EDT inserts a copy of the lines that you specify in range-1 above the range-2 line. New numbers are assigned to the copied lines; the original lines retain their same numbers.

The next example shows how to repeat some headings several times using the COPY command.

```
*TYPE WHOLE
    1       NAME:
    2       ADDR1:
    3       ADDR2:
    4       PHONE:
    5
[EOB]

*COPY 1 THRU 5 TO END
5 lines copied

*TYPE WHOLE
    1       NAME:
    2       ADDR1:
    3       ADDR2:
    4       PHONE:
    5
    6       NAME:
    7       ADDR1:
    8       ADDR2:
    9       PHONE:
   10
[EOB]
```

Since 12 groups of these 5 lines fit on a page, you can use the /DUPLICATE qualifier to make 10 more copies. The /DUPLICATE qualifier tells EDT to perform the COPY operation more than once. The syntax for the /DUPLICATE qualifier is:

```
/DUPLICATE:n
```

Replace the **n** specifier with the number of copies that you want. (The maximum value for **n** is 32767.) To use /DUPLICATE, you must want all the copies inserted at the same location. You have to issue separate COPY commands if you want the copies to be in different parts of your buffer or in different buffers. To insert 10 copies of the headings at the end of the buffer, type this command:

```
*COPY 1 THRU 5 TO END /DUPLICATE:10
5 lines copied 10 times
```

```
*TYPE WHOLE
   1        NAME:
   2        ADDR1:
      .
      .
      .
  56        NAME:
  57        ADDR1:
  58        ADDR2:
  59        PHONE:
  60
[EOB]
```

Notice that in the previous examples, there are no line numbers with decimal fractions. This is because you inserted the text at the end of the buffer. EDT does not have to assign line numbers to fit in between two existing line numbers. Therefore, it numbers the inserted lines consecutively, with increments of 1.

You can use the buffer specifier to copy lines from one buffer to another. If you specify buffer-1, it must already exist in order for EDT to copy lines from it. Otherwise, EDT creates a new buffer with that name and prints the message **No lines copied**.

If you do not specify either buffer-1 or buffer-2 in your COPY command, the operation is made within the current buffer. If you specify only buffer-1, EDT copies the lines in that buffer to the current buffer. Conversely, if you specify only buffer-2, EDT copies the lines in the current buffer to buffer-2. After processing a COPY command, EDT always leaves you in the original buffer or in buffer-2; EDT never stops in buffer-1.

If you do not use a range specifier with buffer-1, EDT copies the entire contents of buffer-1. If you omit the range specifier with buffer-2, EDT inserts the copied text above the first line of buffer-2. If you are using the COPY command to create buffer-2, that buffer will be empty, so there is no need to specify a range.

If you omit both buffer-1 and range-1, be sure you know what the current line is. If EDT is at the end of the current buffer when you issue a COPY command without any location-1 specifiers, you get the message **No lines copied**.

4.13.3 REPLACE

The REPLACE command combines the DELETE and INSERT functions in one command. You can use REPLACE when you need to delete a block of text and want to type new text in that same location. The full syntax is:

```
REPLACE [=buffer] [range] RETURN text CTRL/Z
```

When you use REPLACE, EDT deletes the line or lines specified by buffer and range. As with the DELETE command, EDT prints a message telling you how many lines have been removed. If you supply no specifiers with the REPLACE command, EDT deletes the current line in the current buffer.

As soon as EDT finishes deleting the specified line or lines, it shifts to the insert state. You notice this because the print head or cursor moves to the right, just as it does when you give the INSERT command.

Anything you type, after you have pressed RETURN to process the REPLACE command, is inserted into the buffer. When you finish typing the new text, use CTRL/Z to signal EDT that you want to return to the asterisk prompt (*).

The simplest REPLACE command deletes the current line and shifts to the insert state.

```
*TYPE
   17        This is the current line.

*REPLACE (RETURN)
1 line deleted
             I have just deleted the current line and
             am adding new lines to my text.
             ^Z
   18        This is the next line.

*TYPE 16 THRU 18
   16        This is the line before the current line.
   16.1      I have just deleted the current line and
   16.2      am adding new lines to my text.
   18        This is the next line.
```

4.14 Additional Information about DELETE, INSERT, and REPLACE

Three commands you have already read about – DELETE, INSERT, and REPLACE – have additional syntax forms. When you first saw the DELETE command in this chapter, only the range specifier was mentioned, not the buffer specifier. The same is true of INSERT. In addition, both INSERT and REPLACE have a single-line version that enables you to type the text to be inserted as part of the command line.

4.14.1 DELETE

The full syntax for the DELETE command is:

```
DELETE [=buffer] [range] [/QUERY]
```

The /QUERY qualifier, which allows you to decide which lines in the range or buffer you want to delete, is discussed in the section on Line Mode Qualifiers at the end of this chapter.

When you use only the buffer specifier with DELETE, EDT removes the entire contents of that buffer. But the buffer still exists. The current line becomes the end of buffer ([EOB]) mark in the named buffer.

```
*DELETE =EXTRA
137 lines deleted
[EOB]
```

DELETE and CLEAR have the same effect only when MAIN is the specified buffer. When you use CLEAR to delete buffers other than MAIN or PASTE, EDT eliminates both the contents of the buffer as well as the buffer itself. In the case of the PASTE buffer, DELETE simply deletes the contents; CLEAR not only deletes the PASTE buffer contents, but moves EDT to the MAIN buffer if you issue the command from the PASTE buffer.

4.14.2 INSERT

The INSERT command has two forms. The first enables you to insert as many lines as you want into your text buffer using the same INSERT command. With the second form, you can insert only a single line of text with a single INSERT command.

When the first form of the INSERT command was introduced earlier in this chapter, you saw only its basic syntax: INSERT ... CTRL/Z. Both forms of the INSERT command can take a buffer specifier as well as a range specifier. The full syntax for the two forms is:

```
1.   INSERT [=buffer] [range] RETURN text CTRL/Z
2.   INSERT [=buffer] [range] ;line-to-be-inserted
```

You can use the first form any time you are inserting lines of text into a buffer. The second form, which is limited to inserting a single line of text, is often used in startup command files and EDT macros. It is also useful whenever you have only a single line of text to add. (See Chapter 7 for information on creating startup command files and EDT macros.)

If you use the buffer specifier without a range, EDT inserts the text at the beginning of the buffer, regardless of whether the buffer has anything in it or not. When you specify a range, EDT inserts the text above the line reference. If neither buffer nor range is specified, EDT inserts the text above the current line in the current buffer.

When you use the =buffer specifier with INSERT, EDT moves to the designated buffer and shifts to the insert state. If the buffer is empty or if you want the text inserted at the top of the buffer, you do not need to specify a range. For example:

```
*INSERT =CLOSING
            Sincerely yours,
              .
              .
              .
            Richard L. Thompson
            Director of Design Research
            ^Z
[EOB]
```

The second form of the INSERT command can only take a single line of text. As soon as you press the RETURN key, the command is sent to the computer, so there is no way to enter more than a single line. No CTRL/Z is needed to signal the end of your inserted text because you are not using the RETURN key to start a new line. You must use a semicolon to separate the text from the command word (INSERT) and any specifiers you supply in the command line.

The single line INSERT command is useful when you merely have a small amount of text to insert. In this example you can use the one-line INSERT command to add a city to the list:

```
5       Chicago, IL
6       St. Louis, MO
7       Houston, TX

*INSERT 7 ;Kansas City, KA
*TYPE 5 THRU 7

5       Chicago, IL
6       St. Louis, MO
6.1     Kansas City, KA
7       Houston, TX
```

4.14.3 REPLACE

The REPLACE command, like INSERT, has a second form that enables you to type a single text line as part of your command line. The syntax is:

```
REPLACE [=buffer] [range] ;line-to-be-inserted
```

The text to be inserted is separated from the command and specifiers by the semicolon. No CTRL/Z is needed.

As with the usual form of REPLACE, if you specify no buffer or range, the current line is replaced by the inserted text. If you specify a buffer without a range, the entire contents of the buffer are deleted and replaced with the inserted text.

The single line form of the REPLACE command is especially useful in performing substitutions for entire lines because you have to type only the new text. The following example shows how to substitute one name for another in a list:

```
12        Accounting Department
13           Bonnie Hutchins
14           Marilyn Dunten
15           Mary Cartwright
16

*REPLACE 15 ;Jackie Tenney
1 line deleted
16

*TYPE 12 thru 15

12        Accounting Department
13           Bonnie Hutchins
14           Marilyn Dunten
15           Jackie Tenney
```

4.15 Commands That Use External Files — INCLUDE, PRINT, WRITE

When you use the INCLUDE, PRINT, or WRITE commands, EDT can reach files in your directory (or in another directory that you have access to) without interrupting your editing session. INCLUDE brings a copy of an external file into your EDT session. PRINT and WRITE enable you to create an external file with material from your current editing session.

These commands are useful in any editing mode. They can save hours of work by eliminating the need to retype the same material in several different files. Consider these applications:

1. You have to send a letter to 10 people whose names and addresses are in a file. The letter is basically the same for each person but requires some minor editing to personalize the contents. Use EDT to type the basic letter. Then use the INCLUDE command to put a copy of the address list in a separate buffer in your EDT session. You can now use the MOVE command to put the first address at the appropriate spot in your letter file. After you finish the edits necessary to personalize the letter for the first individual, use the WRITE command to create a file containing that copy of the letter. You are ready for the second letter. Delete the old address and move the next one into place. Edit the letter so that it reflects the next recipient and

then use the WRITE command to copy that version to another external file. Repeat the process until all 10 letters are finished.

2. When you write computer programs, you can often use elements from other programs in the one you are currently creating. Use EDT to look at the old program. With the WRITE command, copy the lines you want for your new program into a file in your directory. Now call up EDT to work on the new program and use the INCLUDE command to bring in the lines you need from the old program.

   Or, while you are working on the new program, use INCLUDE to copy the old program into another buffer. Shift to that buffer to delete all the unwanted lines. Then use MOVE to put the remaining lines in the new program.

3. If you are in the middle of an EDT session and need to get some information or text from another file, use INCLUDE to copy the other file into a separate buffer. Once you find the item you need, you can resume editing in your original buffer.

4.15.1 INCLUDE

The INCLUDE command copies the entire contents of an external file to a specified location in your editing session. The full syntax is:

```
INCLUDE file-specification [=buffer] [range]
```

File-specification is the file that you want copied into your EDT session. You must supply this information. If you supply neither the buffer nor range specifier, EDT places the copied text above the current line in the current buffer. To include the text in a new buffer, simply type the name of the buffer after the file specification. EDT moves to the end of the specified buffer.

Range refers only to a single line in your current editing session. It does not refer to a portion of the file being included. If you do not need the entire contents of the file you have specified, put it in a separate buffer and use the DELETE command to discard the unwanted lines, or use the MOVE or COPY command to gather only the lines you need.

This example uses the closing for a letter that has been stored in a separate file. When you have finished typing the body of a letter, insert the contents of the file CLOSING.LET at the end of the current buffer.

```
                look forward to seeing you next week.

[EOB]

*INCLUDE CLOSING.LET END
*TYPE -"week" THRU END
   76           look forward to seeing you next week.
   77
   78           Sincerely yours,
                      .
                      .
                      .
   91           HCJ/mqt
[EOB]
```

4.15.2 WRITE

The WRITE command makes a copy of the specified text and puts it in an external file. Since WRITE only copies your text, it has no effect on the contents of your editing session. The new file can be created in the current directory or in another directory that you have access to. If you use a directory name as part of your file specification, that directory must exist and you must have access to it in order for EDT to create a file there. EDT creates files with the WRITE command, but it cannot create directories.

The WRITE command uses the /SEQUENCE qualifier to put sequence numbers in the external file. For information on how to use this qualifier with the WRITE command, see the section in Chapter 7 on EDT line numbers.

The full syntax for the WRITE command is:

```
WRITE file-specification [=buffer] [range]
      [/SEQUENCE [:initial [:increment] ] ]
```

You must supply the output file specification with the WRITE command. If you do not use a directory name in the file specification, EDT creates the file in the current directory.

The buffer and range specifiers identify the text to be copied to the external file. If you specify a buffer without any range, EDT copies the entire buffer. EDT does *not* move to the buffer specified with the WRITE command.

If you omit the buffer specifier, the range specifier refers to lines in the current buffer. When neither the buffer nor range specifier is given, EDT copies the entire contents of the current buffer to the specified file.

EDT lets you know that it has created the output file by printing the full file specification after your WRITE command has been processed.

```
35        These are the attendees from our group:
36
37        Jane Hillyar, Sales
38        Martin Schulman, Sales
39        Adrian Balsam, Sales
40        Jerry Du Val, Development
41        Harriet Schwartz, Development

*WRITE MEETING.DAT 37 THRU 41
DISK$USER:[JAMISON]MEETING.DAT;1 5 lines
```

When you issue the WRITE command directly from a screen mode, you can use a select range to designate the material you want copied. Create the select range using the keypad SELECT function or the nokeypad SEL command. Remember that your select range can consist only of entire lines in order for you to use SELECT as the range specifier.

```
Command:  WRITE TABLE6.DAT SELECT
DISK$USER:[PETERS]TABLE6.DAT;1  27 lines
```

4.15.3 PRINT

The PRINT command copies text to an external file. Its syntax is similar to the WRITE command, but without the /SEQUENCE qualifier. The full syntax is:

```
PRINT file-specification [=buffer] [range]
```

There is no /SEQUENCE qualifier because EDT always copies the existing EDT line numbers to the external file with the PRINT command. The EDT line numbers become part of the text in the external file and they can be edited in subsequent editing sessions. In addition, EDT adds a form feed and two blank lines every 60 lines so that the file can be conveniently printed on a line printer.

You might want to use the RESEQUENCE command to renumber your text before issuing the PRINT command because the EDT line numbers become part of the text that is copied to the external file. The RESEQUENCE command without the /SEQUENCE qualifier renumbers your buffer lines starting with 1 and incrementing by 1. For more information on both the RESEQUENCE command itself and its use of the /SEQUENCE qualifier, see the section on the RESEQUENCE command later in this chapter.

EDT does not display a file specification message after processing a PRINT command. If you are using PRINT in line mode, EDT displays the asterisk prompt (*) after it has processed the command.

```
*PRINT YESNOSUB.BAS 610 THRU 840
*
```

You must supply a file specification with the PRINT command. If you give no directory name in the file specification, EDT creates the file in the current directory. If you do specify a directory, that directory must exist and you must have access to it in order for EDT to create a file there.

When you do not supply a buffer specifier, EDT assumes the current buffer. If you omit the range specifier but include a buffer specifier, EDT copies the entire buffer. If you include neither a buffer nor range specifier, EDT copies the entire current buffer. EDT does *not* move to the buffer specified with the PRINT command.

4.16 Line Numbering Your Text — RESEQUENCE

The RESEQUENCE command renumbers the EDT line numbers in the current buffer or the specified buffer. You can use the RESEQUENCE command to simplify the line numbers in a buffer, to find out how many lines you have in a buffer, or to reorder the EDT line numbers before they are converted into text by the PRINT command.

The syntax for RESEQUENCE is:

```
RESEQUENCE [=buffer] [range] [/SEQUENCE[:initial[:increment]] ]
```

Range can be a single line, but is generally a group of lines. If you omit the buffer specifier, EDT renumbers the lines in the current buffer. When you give only a buffer specifier, EDT moves to that buffer and renumbers every line. If you specify an empty buffer, EDT moves to that buffer and prints the message **No lines resequenced**. When you supply neither buffer nor range specifier, EDT resequences the entire current buffer.

If you are renumbering an entire buffer and do not use the /SEQUENCE qualifier, EDT starts with 1 for the first line and increments the line numbers by 1 until the end of the buffer is reached. If you have 37 lines in your buffer, the first line will be numbered 1 and the last 37.

The /SEQUENCE qualifier serves several purposes. It is used not only with the RESEQUENCE command, but also with EXIT and WRITE. However, when used with EXIT and WRITE, the qualifier has different defaults and limitations for the :initial and :increment specifiers. Information on the effects of /SEQUENCE with EXIT and WRITE appears in the section on Using EDT Line Numbers in Chapter 7.

When you use the /SEQUENCE qualifier with the RESEQUENCE command, there are several things to be aware of. First of all, the /SEQUENCE qualifier is optional. If you omit it, EDT resequences the specified lines using an initial value of 1 and an increment of 1 if it can.

If you omit the /SEQUENCE qualifier and the :initial specifier from the RESEQUENCE command line, you must want either to resequence the entire buffer or to resequence a range that begins with a line having a number less than or equal to 1. Otherwise EDT prints a warning message. For example, suppose you want to resequence these lines:

```
12.8      504-N-374-4786
12.9      504-N-438-5479
13        504-N-578-8763

*RESEQUENCE 12.8 THRU 13
Range specified by /SEQUENCE would cause duplicate or nonsequential line numbers
```

Even though you do not use the /SEQUENCE qualifier, EDT assumes that you want /SEQUENCE:1:1. Either there is already a line with number 1 in the file or conflicts with the increment numbers would occur. If you type just the /SEQUENCE qualifier with no initial or increment value, the same message appears. Thus, using the /SEQUENCE qualifier without its specifiers in the RESEQUENCE command line has the same effect as using RESEQUENCE with no qualifier. However, when you add an initial value of 13 to your command, EDT has no trouble resequencing the lines.

```
*RESEQUENCE 12.8 THRU 13 /SEQUENCE:13
3 lines resequenced

*TYPE 13 THRU END

   13        504-N-374-4786
   14        504-N-438-5479
   15        504-N-578-8763
[EOB]
```

When you want to resequence some lines in the middle of your text, unless there is a sufficient gap in your current line numbers, EDT resequences from the first line in the range to the end of the buffer, if necessary. Even if you give only five lines in your range, EDT might need to resequence 27 lines, for instance, in order to avoid overlapping or duplicating line numbers. In this case, EDT tells you how many lines were actually renumbered, for example, **27 lines resequenced**.

This example resequences a list of names starting with the second line in the buffer.

```
1       Robert Hargraves
1.2     Ilse Huston
1.3     Michelle Louzier-O'Neill
2       Norman Saunders
3       JoAnn Strathmeyer
4       William Way

*RESEQUENCE 1.2 THRU 2 /SEQUENCE:2:1
5 lines resequenced

*TYPE 1 THRU "way"

1       Robert Hargraves
2       Ilse Huston
3       Michelle Louzier-O'Neill
4       Norman Saunders
5       JoAnn Strathmeyer
6       William Way
```

4.17 Formatting Text — FILL

Line mode has two commands that format text: FILL and TAB ADJUST. The TAB ADJUST command is used to indent lines or blocks of lines for outlines, indented programs, or other layered text. This command is discussed in Chapter 7 in the section on EDT's Tabbing Facility.

The FILL command arranges a block of text so that as many whole words as possible fit on each line. Whenever necessary, EDT replaces line terminators with spaces or spaces with line terminators to achieve a filled block of text.

The syntax for the FILL command is:

```
FILL [=buffer] [range]
```

To use the FILL command, first decide which group of lines you want EDT to reformat. Give those as the range or buffer specifier with the command. If you give no specifiers, EDT assumes that you have an active select range set by either the keypad SELECT function or the nokeypad SEL command. In order to have EDT reformat the entire current buffer, you must include the current buffer name in the FILL command line, for example FILL =MAIN.

When you give only a buffer specifier, EDT reformats the entire buffer. A range specifier with no buffer specifier refers to lines in the current buffer.

EDT uses the current SET WRAP value or the SET SCREEN width to determine the maximum number of characters that can be put on a line. Both SET SCREEN and SET WRAP work with hardcopy terminals as well as screen terminals. EDT's default line length is taken from the operating system, but remember that the maximum line length for some terminals is 80 characters.

EDT has no default SET WRAP value. SET NOWRAP is the default.

If you use FILL without resetting the line length, EDT puts as many whole words as it can on each line without exceeding the system set line length. By changing the SET SCREEN value or establishing a SET WRAP value, you can change the line length that EDT uses with the FILL command. If you have a SET WRAP value established, EDT uses that number rather than the SET SCREEN width to determine the line length. IF SET NOWRAP is in effect, EDT sets the line length for filling at one less than the current SET SCREEN width.

This example shows the text in its original state. To reformat the lines so that the maximum line length is 50 characters, first issue the SET WRAP 50 command. Then use FILL to reformat the text.

```
    1       Acronyms are words formed from the initial letters
    2       or parts
    3       of compound terms.
    4       Mnemonics are memory devices,
    5       often formed by combining letters to form word-like
    6       expressions.
    7       The first time you use an acronym or mnemonic,
    8       spell out the word or phrase
    9       and follow it with the shortened form in parentheses.
[EOB]

*SET WRAP 50
*FILL 1 THRU 9
*TYPE WHOLE

    1       Acronyms are words formed from the initial
    2       letters or parts of compound terms. Mnemonics are
    3       memory devices, often formed by combining letters
    4       to form word-like expressions. The first time you
    5       use an acronym or mnemonic, spell out the word or
    6       phrase and follow it with the shortened form in
    7       parentheses.
[EOB]
```

Notice that EDT renumbers the lines. If filling the text results in more lines being created, EDT uses line numbers with decimal fractions for the additional lines. You can use the RESEQUENCE command to change the line numbers if you need to.

4.18 TAB

Line mode can use the TAB key to insert horizontal tab characters into your text. EDT has preset tab stops every eight characters. When you use the TAB key while inserting text, EDT inserts a tab character (control I) into your text. This character moves subsequent text to the next preset tab stop.

Because of the line number field, the location of tab stops in line mode appears differently when you are in line mode than it does when you display the text in a screen mode or outside of EDT. You can see the actual effects of using the TAB key during your EDT session if you use the SET NO NUMBERS command. (SET NUMBERS is the default.)

With SET NUMBERS in effect, the tab stops appear to be at columns 4, 12, 20, 28, and so on. However, they are actually at columns 8, 16, 24, 32, and so on. If you give the SET NONUMBERS command, EDT no longer displays the line number field. Thus, you see the text displayed accurately on your paper or screen while still in line mode.

The next example uses a guide line to show the column positions in the tabbed line. The first two lines are as they appear in EDT.

```
1        12345678123456781234567812345678123456781234567812345678
2        A   B       C       D       E       F       G
```

When you exit from EDT and print the lines at your terminal or on a line printer, they appear this way:

```
12345678123456781234567812345678123456781234567812345678
A       B       C       D       E       F       G
```

4.19 Using Several Line Mode Commands on a Single Line

EDT allows you to enter more than one line mode command on a single command line. You use the semicolon (;) to separate the individual commands from each other.

You can join as many as 19 line mode commands together with semicolons before pressing RETURN, but the total number of characters that EDT can read in a single command line is 255. In most instances you will not want to type more commands than you can fit on your paper or screen. However, EDT will wrap the command line when it reaches the righthand side and allow you to continue typing more commands.

Some commands can only be used at the end of a multicommand line. These include CHANGE and the commands that use the semicolon in one version of their syntax – the single-line form of the INSERT and REPLACE commands.

You can use multicommand lines to issue several SET commands on a single line, add a TYPE or FIND command after deleting or inserting text, or move to a new buffer after completing an operation. You might find it useful to add a **FIND =MAIN .** after using COPY or MOVE so that EDT returns to the original buffer.

These examples show multicommand lines. EDT does not care if you use spaces before or after the semicolons, but spaces make it easier for you to distinguish the individual commands.

```
*SET SEARCH EXACT ; SET WRAP 70 ; SET TAB 5 ; SET QUIET
```
This is a series of SET commands.

```
*COPY =TEST1 1 THRU 10 TO =TEST2 153 ; FIND =MAIN .
```
After EDT completes the COPY operation, it processes the FIND command to return to the most recent current line in the MAIN buffer.

```
*DELETE REST ; FIND =PASTE
```
First EDT deletes the remaining lines in the current buffer. Then it moves to line 1 of the PASTE buffer.

```
*MOVE 67 THRU 183 TO =EXTRA ; SHOW BUFFER
```
EDT moves some lines to the EXTRA buffer and then lists all the buffers in your EDT session.

```
*SET NOKEYPAD ; CHANGE
```
EDT sets the screen mode to nokeypad and then shifts to that mode.

4.20 QUALIFIERS in Line Mode

Line mode has seven qualifiers. Whenever you use one, it must be preceded by a slash (/) to signal EDT that you have typed a qualifier. Line mode qualifiers always come at the end of the line mode command line, following any specifiers that you might include with the command.

The qualifiers can be abbreviated to from one to three letters. Table 4-5 lists the qualifiers and the commands that use them. The underscored letters indicate the abbreviations.

Table 4-5: Line Mode Qualifiers

Qualifier	Line Mode Commands
/BRIEF[:n]	SUBSTITUTE, TYPE
/DUPLICATE:n	COPY
/NOTYPE	SUBSTITUTE
/QUERY	COPY, DELETE, MOVE, SUBSTITUTE
/SAVE	EXIT, QUIT
/SEQUENCE[:init.[:inc.]]	EXIT, RESEQUENCE, WRITE
/STAY	TYPE

Some of the qualifiers are discussed elsewhere in the manual. The /DUPLICATE qualifier, which enables you to copy a group of lines several times in the same location, is described with the COPY command. The /SAVE qualifier is discussed in Chapters 2 and 7 in the sections on using journal files, because this qualifier causes EDT to save a copy of the journal file. Details about the /SEQUENCE qualifier are given in this chapter in the section on the RESEQUENCE command. Information on using the /SEQUENCE qualifier with PRINT and WRITE appears in the line numbering section in Chapter 7.

The remaining qualifiers, /BRIEF, /NOTYPE, /QUERY, and /STAY are described in the next sections.

4.20.1 /BRIEF

The /BRIEF qualifier limits the number of characters that EDT displays when you use the SUBSTITUTE or TYPE command. When you issue the SUBSTITUTE command, you do not always need to have EDT display every character on each line where a substitution occurred. Similarly, you might only need to see the first few characters of the lines displayed by a TYPE command.

The syntax for /BRIEF is:

```
/BRIEF[:n]
```

The /BRIEF qualifier instructs EDT to print only the first **n** characters of the line. If you do not include the **n** specifier, EDT uses a default value of 10. This first example has EDT display the first 15 characters of each line on which a substitution takes place.

```
*SUBSTITUTE/1983/1984/ 5 THRU 35 /BRIEF:15
    6       June 13, 1984
   13       happened on tha
   16       when the event
   23       final week in J
4 substitutions
```

The next example omits the **n** specifier with /BRIEF. EDT prints the first 10 characters of each line in the range.

```
*TYPE 42 THRU 45 /BRIEF
   42       FAIRMONT,
   42.1     MEDFORD, O
   42.2     SACRAMENTO
   43       DETROIT, M
   44       READING, P
   45       ABILENE, T
```

4.20.2 /NOTYPE

The /NOTYPE qualifier is used only with the SUBSTITUTE command. When you use /NOTYPE, EDT does not display any of the lines in which substitutions took place. This qualifier is especially useful if you are doing a substitution involving many lines.

```
*SUBSTITUTE/DEC/DIGITAL/ WHOLE /NOTYPE
136 substitutions
```

Notice that there are four slashes in the command line. The slashes surrounding **DEC** and **DIGITAL** are delimiters. You can change these delimiters to some other punctuation mark (except % or _) if you need to. The slash preceding the qualifier name **NOTYPE** must always be a slash. It is a signal, not a delimiter. The following example shows a SUBSTITUTE command with different delimiters because the slash mark appears in one of the strings:

```
*SUBSTITUTE'I/O'input-output' WHOLE /NOTYPE
```

4.20.3 /QUERY

The /QUERY qualifier allows you to decide whether you want EDT to perform the command operation on each line in the specified range or buffer or only on some of the lines. Use this qualifier when you want EDT to process a COPY, DELETE, MOVE, or SUBSTITUTE command on most of the lines in the range or buffer, but not all. With the /QUERY qualifier in effect, EDT pauses after printing each line and displays a question mark prompt (?). You respond with one of the following:

Y	(YES)	Perform the operation on this line.
N	(NO)	Skip over this line. Leave it as is.
A	(ALL)	Perform the operation on all the remaining lines without stopping.
Q	(QUIT)	Stop processing the command now.

With the COPY command, the /QUERY qualifier causes EDT to pause at each line in the range being copied to find out if you want that particular line duplicated in the new location. Suppose you have a range of ten lines in your COPY command, but only want nine of them copied. Respond with Y (YES) until you reach the unwanted line. Use N (NO) in response to the /QUERY prompt (?) for the line you do not want copied. Then you can use A (ALL) to have EDT copy the remaining lines, without further prompting.

You can use the /QUERY qualifier with DELETE or MOVE in the same way. If you want to delete 23 lines from a 25 line range, and are not sure of the numbers of those lines you want to keep, use the /QUERY qualifier with your DELETE command. Similarly, with MOVE you can respond with Y (YES) for the lines you need moved, and have EDT leave the other lines in their original location.

The /QUERY qualifier is very useful with SUBSTITUTE. For example, suppose you need to change the year from 1983 to 1984 at most places in your text, but not all. Using the /QUERY qualifier, you can have EDT display the lines containing the search string and prompt you after each line. Then you can decide whether or not to make the substitution on that line. If there are two or more instances of the search string on a line, EDT prompts you separately for each occurrence of the string.

If you have used the /QUERY qualifier with a command, you can have EDT stop further processing of that command by typing Q (QUIT) in response to the /QUERY prompt. Lines that have already been processed by EDT are not affected by the Q (QUIT) response. EDT is now ready to receive another command.

The following example demonstrates the /QUERY qualifier with a SUBSTITUTE command. You can change some years in the text from 1983 to 1984, but do not have to change all of them.

```
*SUBSTITUTE/1983/1984/WHOLE /QUERY
    1       DATE:  January 3, 1983
?RETURN
Please answer Y(es), N(o), Q(uit), or A(ll)
?N
    2.01    We are planning the 1983 convention.
?Y
    2.01    We are planning the 1984 convention.
    2.03    in July 1983, October 1983, January
?N
    2.03    in July 1983, October 1983, January
?N
    2.031   1983, and April 1983.
?A
    2.031   1984, and April 1984.
3 substitutions
```

Notice that if you do not type a letter in response to the question mark prompt, EDT prints a reminder message indicating the valid answers for the /QUERY prompt. In line 2.03 there are two instances of the string. EDT prompts for each one. In line 2.031, you can respond A (ALL) to complete the job without any further prompts. After making the substitutions, EDT tells you how many substitutions were made.

4.20.4 /STAY

The /STAY qualifier is used only with the TYPE command. It affects the location of the current line after EDT has processed the TYPE command. The /STAY qualifier enables you to look at other text in your editing session without leaving your current position. You can even have EDT display lines from another buffer without leaving the current line in the current buffer.

Normally when EDT processes a TYPE command, the first line of the range or buffer specified in the command becomes the new current line. If you want EDT to remain at the line that is current when you issue the TYPE command, use the /STAY qualifier. The next example displays the range 3 through 9 and then performs a substitution on the current line, which is still line 25.

```
*TYPE .

   25        and to all my sincerest appreciation.

*TYPE 3 THRU 9 /STAY
    3        DATE:   December 26, 1985
    4
    5        TO:     People Who Sent Me Christmas Presents
    6
    7        FROM:   Andy Crony
    8
    9        RE:     Thanks!

*SUBSTITUTE/./!!/
   25        and to all my sincerest appreciation!!
1 substitution
```

Notice that when you use the SUBSTITUTE command, EDT is still at line 25, where you want the substitution to occur.

This example uses the /STAY qualifier to remain at the current line in the current buffer, while EDT displays the contents of the JULY buffer.

```
 2746        for the next several years.  Of course, after

*TYPE =JULY /STAY

    1        Projected Figures for July 1986
    2
       .
       .
       .
[EOB]

*TYPE .
 2746        for the next several years.  Of course, after
```

Chapter 5
Nokeypad Editing

5.1 Introduction

This chapter describes the various elements of nokeypad editing and shows how to use nokeypad commands. The first section gives definitions of terms that are used in describing nokeypad editing. The next section deals with basic elements of nokeypad editing, such as cursor position and command specifiers, including the entity specifier. The section on nokeypad commands includes information on entering several commands on the same command line. This feature is especially useful when you are creating key definitions for keypad mode. (Keypad key definitions are based on nokeypad commands.)

Nokeypad editing is the second EDT screen mode for use with VT100-type and VT52 terminals. Nokeypad mode is a character-oriented mode, as is keypad mode. Of the two, keypad is easier to learn and use, but nokeypad editing has more commands and entity specifiers. When you combine the two modes through keypad key definitions and direct use of nokeypad commands while in keypad mode, you have an editor with a wide variety of functions and capabilities.

Nokeypad editing uses a cursor to show the position of EDT in the text you are editing. The cursor moves between the text displayed on the screen and the command line. Text is displayed on the top 22 lines of the screen. The commands you enter appear on the 23rd line of the screen.

When you use the arrow keys to move the cursor, the command line remains blank and the cursor moves around in the text buffer. As soon as you start typing a command, the cursor shifts to the command line (below the text) and remains there until you press RETURN to process the command. Once the command has been processed, the cursor returns to the screen to show the new position of EDT in the text.

You can give several nokeypad commands in a single command line. It is this feature that enables you to perform several editing functions with only one or two keystrokes in keypad mode, which uses nokeypad commands to construct the keypad editing key definitions. (To find out how to define keys for keypad mode, see Chapter 7.) For example, you can move EDT to a new position in the text, delete several words, and then move on to another string in a single command line.

```
4L 2W D3W "1985"
```

If your terminal can handle an EDT screen mode, you will most likely use keypad editing. Nokeypad editing is primarily for:

- Creating key definitions for keypad editing
- Working at low data transmission rates

- Specifying entities not available in keypad mode
- Using commands not available in keypad mode
- Using hardcopy change mode

Keypad key definitions are used only in keypad mode. However, the definitions are based on nokeypad command syntax. The better you understand nokeypad commands and specifiers, the more keypad functions you can create.

Some people who have terminals operating at low data transmission rates prefer to use nokeypad mode. See Appendix C for specific information on using screen terminals at these low data rates.

Nokeypad mode has more commands and specifiers than keypad mode. For example, keypad editing relies on four entities – character, word, line, and page; nokeypad mode has 22 (see Table 5-1). Some commands, such as those that shift text horizontally on your screen, are available only in nokeypad mode. In addition, you can join several nokeypad commands on a single nokeypad command line.

When you use EDT's keypad definition facility, you can create key definitions to use the additional nokeypad commands and specifiers during your keypad editing session. The combination of the two screen modes enables you to customize your editing work.

Hardcopy change mode is a special EDT facility that allows you to use nokeypad commands on a terminal that EDT does not support for screen editing. A brief description of how to use this editing mode appears in Chapter 7.

5.2 Definitions of Terms Used in Chapter 5

These terms are used in describing nokeypad editing. Definitions of other EDT terms appear in Chapter 2.

current position

The location of the cursor in the text before you type a command.

delimiter

A punctuation character used to set off strings in commands. In EDT certain types of strings require quotations marks (either single – ', or double – ") as delimiters.

syntax

The structure of a command statement.

5.3 Basics of Nokeypad Commands

Nokeypad editing uses commands that consist of either English words or abbreviations. The three general forms for nokeypad commands are:

```
command

[+|-][count]command

[+|-][count]command[+|-][count]entity[=buffer]
```

As in line mode, you press the RETURN key to have EDT process the commands. You can join several commands on a single command line and process all of them with one RETURN.

Notice that there are no spaces in nokeypad command syntax. For example, the single command CUT2SEN=EXTRA has no spaces. When you type two commands on the line, such as 3W D-C, you can separate them with a space. However, spaces between commands are optional.

The syntax statements show several specifiers. Specifiers enclosed in square brackets ([ ]) are optional. Notice that the entity specifier is *not* optional in the third sample command line.

In addition to the nokeypad commands, you can use the four arrow keys to move the cursor around on the screen and the DELETE key to delete characters from your text as well as from the command line.

5.4 Nokeypad Specifiers

The nokeypad specifiers are:

buffer	count	number	string
character	entity	+|–(sign)	

Two specifiers are used only with a single command: the character specifier with circumflex (^) and the number specifier with ASC. These specifiers are discussed with their respective commands.

5.4.1 The Buffer Specifier

Nokeypad mode uses the buffer specifier with only three commands:

APPEND	CUT	PASTE

The buffer name, which must follow the entity specifier, is always preceded by an equal sign (=) signal. There are no spaces either before or after the equal sign.

```
APPEND␣PAR=SAVE
```

If you do not specify a buffer name with these three commands, EDT uses the PASTE buffer, the default.

As with line mode, whenever you specify a buffer, EDT creates that buffer if it does not already exist. The line mode command SHOW BUFFER tells you the names of the buffers that currently exist in your editing session and points to the current buffer. Unlike the line mode buffer specifier, the nokeypad buffer specifier does *not* move EDT to the named buffer. EDT remains in the same buffer it was in before you issued the command. You must use line mode commands to move from buffer to buffer.

5.4.2 The Count Specifier

There are two ways to use the count specifier: (1) to indicate how many times a command is to be processed and (2) to indicate how many entities the command is to affect. When a command can take only one count specifier, the specifier always precedes the command and always tells EDT how many times to repeat the command. The maximum value for the count specifier is 32767.

When commands take two count specifiers, the first one is the repeat count, that is, the number of times the command is to be repeated; the second is the entity count, the number of characters, words, lines, or whatever that are to be affected by the command.

In most cases, a repeat count or an entity count will produce the same results. For example, the commands D3W and 3DW both delete three words.

5.4.3 The Sign (+ | –) Specifier

The sign specifier overrides the current direction of your EDT session, but only for the command that it is part of and only for some entities. The signs do not change EDT's direction. The plus sign (+) tells EDT to process the command in the forward direction, the default. The minus sign (–) tells EDT to work backward toward the beginning of the buffer. To change EDT's direction for a series of commands, use the ADV (advance) or BACK (backup) command.

Most entities that contain the B (beginning) or E (end) prefix are not affected by the sign specifier. They are also not affected by ADV or BACK. The B prefix always focuses EDT to the left; E focuses EDT to the right. The exception to the rule is EL (end of line), which is affected by the plus or minus sign as well as by ADV or BACK.

Generally, it makes no difference if you put the sign specifier before the repeat count or the entity count. The commands –3DC and 3D–C produce the same results. However, with the TADJ (tab adjust) command, the effects are different. –TADJ3L does not result in the same change to your text as TADJ–3L.

5.4.4 The String Specifier

The string specifier is used with three nokeypad commands:

S (substitute) SSEL (search and select) XLATE

The S command can take two string specifiers – a search string and a substitute string. The string used with SSEL is always a search string. The XLATE string is a parameter that EDT passes to a program when running callable EDT from the VAX/VMS operating system. For information on XLATE, see Appendix D.

When you use the string specifier as a search string, EDT stores the string in the search buffer. Similarly, if you supply a substitute string with the S command, EDT stores that string in the substitute buffer. Strings in S and SSEL commands are set off from the command letters by delimiters. For SSEL, you must use quotation marks, either single (') or double ("). You can use any nonalphanumeric character as a string delimiter for the S command.

The important thing to remember about the string specifier is that it differs from the string entity. A string specifier includes those characters that constitute the string itself. The string entity includes the initial cursor character and those characters that lie *between* the cursor and the string. Suppose you have the following line in your text with the cursor located between the words **today** and **or**:

```
Are you going to the movies today□or the concert?
```

If the string specifier is the question mark (?), then the question mark will be affected by the command.

```
S/?/??/

     Are you going to the movies today or the concert??
```

However, if you use the string entity with your command, different characters are affected:

```
D'?'

     Are you going to the movies today?
```

The string specifier was the question mark (?). The string entity was SP **or the concert.**

5.4.5 The Entity Specifier

An entity is a unit of text that EDT recognizes when processing commands. Nokeypad mode has 22 entities, which are described in Table 5-1. (See the section on SET ENTITY in Chapter 6 for more information on how EDT defines four of these entities: WORD, SENTENCE, PARAGRAPH, and PAGE.)

Table 5-1: Nokeypad Entities

Entity	Description
C	Character – a single character. Characters include: letters, digits, punctuation marks, and control characters.
W	Word – a string of characters bounded by a delimiter character. The default boundary delimiters are: space, horizontal tab, line feed, vertical tab, form feed, carriage return, and line terminator. Each boundary character except space is considered to be a word in itself, unless SET WORD NODELIMITER is in effect. A space following a word is considered to be part of the word.
BW	Beginning of word – the string of characters starting with the character to the left of the cursor and extending to the left until the beginning of the word is reached.
EW	End of word – the string of characters starting at the cursor and extending to the right until the end of the word is reached.
L	Line – a single line of text starting just after a line terminator and encompassing all characters up to and including the next line terminator.
BL	Beginning of line – the string of characters starting from the cursor and extending to the left until the beginning of a line is reached. When the cursor is already at the beginning of a line, the unit of text extends to the beginning of the previous line.
EL	End of line – the string of characters starting at the cursor and extending to the end of the line, up to but not including the line terminator. When the cursor is already at the end of a line, the unit of text extends to the end of the next line.
NL	Next line – the string of characters from the cursor to the beginning of the next line. The ending line terminator is included in the string.
SEN	Sentence – a string of characters bounded by a delimiter character and at least one space or line terminator. The default boundary delimiters are: period (.), exclamation point (!), and question mark (?).

(continued on next page)

Table 5-1: Nokeypad Entities (Cont.)

Entity	Description
BSEN	Beginning of sentence – the string of characters starting from the cursor and extending to the left until the beginning of the sentence is reached.
ESEN	End of sentence – the string of characters starting at the cursor and extending to the right until the end of the sentence is reached. The boundary mark is not included.
PAR	Paragraph – a string of characters bounded by a delimiter character or delimiting string. The default boundary delimiter is two line terminators in succession.
BPAR	Beginning of paragraph – the string of characters starting from the cursor and extending to the left until the beginning of the paragraph is reached.
EPAR	End of paragraph – the string of characters starting at the cursor and extending to the right until the end of the paragraph is reached. The paragraph boundary is not included. EPAR cannot be used twice in succession, nor can it be used with a repeat or entity count.
PAGE	A string of characters bounded by a delimiter character or delimiting string. The default boundary delimiter is the form feed.
BPAGE	Beginning of page – the string of characters starting from the cursor and extending to the left until the beginning of the page is reached. If there are no page markers, BPAGE is the same as BR.
EPAGE	End of page – the string of characters starting at the cursor and extending to the right until the end of a page is reached. The boundary mark is not included. If there are no page markers, EPAGE is the same as ER. EPAGE cannot be used twice in succession, nor can it be used with a repeat or entity count.
SR	Select range – the string of contiguous characters between the position marked by the SEL command and the next cursor position. When no select range is active, the cursor is at the current search string, and the repeat count is not greater than 1, that string becomes the select range. When these conditions do not exist, SR becomes a single character but only for use with the CHGC, CHGL, and CHGU (change case, change case lower, and change case upper) commands.
"string" 'string'	All the characters between the previous cursor position and the specified string. The string must be enclosed with either single (') or double (") quotation marks. The string characters themselves are not affected by the command.
BR	Beginning of range – the string of characters starting with the character to the left of the cursor and moving to the left to the beginning of the current buffer.
ER	End of range – the string of characters starting at the cursor and moving to the right to the end of the current buffer.
V	Vertical – the string of characters starting at the cursor and moving to the identical column position of the line above or below, depending on the current or specified direction. Vertical always includes a line terminator. If the next line has fewer characters than the current column, EDT moves to the last character in the line. If there is a horizontal tab, EDT moves left of the proper column to the tab character.

The sign specifier and the ADV and BACK commands have no effect on entities with the B or E prefix, such as BW, ESEN, and BPAGE, except for EL.

It is important to remember which entities include the ending boundaries and which entities affect the character the cursor is on. In general, entities beginning with B or E do not include the boundary marker. When the entity is a string of characters to the left of the cursor, the current cursor character is not included in the entity. When the entity involves characters to the right of the cursor, the cursor character is included in the entity.

The following examples show how the D (delete) command works with the three different word entities. In the first example EDT deletes the entire word **Fourth**:

```
The Fourth of July

DEW

The of July
```

If you use the BW entity with the same example, EDT deletes the word **The**. Notice that the cursor character is not deleted.

```
The Fourth of July

DBW

Fourth of July
```

The command DW has the same effect as DEW when the cursor is at the beginning of a word. However, when the cursor is in the middle of the word, the effect is different.

```
The Fourth of July

DW

The of July
```

EDT deletes the entire word with DW even when the cursor is in the middle of the word. With DEW, only the cursor character and the characters up to the beginning of the next word are deleted. Using the same initial cursor position, give the DEW command.

```
The Fourth of July

DEW

The Fourof July
```

The next examples compare DBW with –DW, first with the cursor on the space between words, then with the cursor on the last letter of a word. The –DW command deletes the word to the left of the cursor including the space boundary.

```
The Fourth of July

-DW

Fourth of July
```

With DBW, EDT deletes the word **Fourth**, but not either of the surrounding spaces. Now there are two spaces between the words **The** and **of**.

```
The Fourth of July

DBW

The of July
```

In the next example, the cursor is on the last character in the word **Fourth**. The effect of –DW is the same with the cursor in this position as it was with the cursor between the words **Fourth** and **of**.

```
The Fourth of July

-DW

Fourth of July
```

(If you use D–W instead of –DW, the effect is the same.)

However, with the cursor on the **h** in **Fourth**, DBW deletes only those characters to the left of the cursor, up to, but not including, the space, which is a word boundary.

```
The Fourth of July

DBW

The h of July
```

The nokeypad "move" command uses the entity specifier with no command word. To use the "move" command, you simply type the entity specifier with or without a sign or count specifier. (Do not type the word "move" or the quotation marks.) The next examples use the "move" command with various line entities to show the differences in cursor position between L, BL, EL, and NL.

```
One for the money.
Two for the show.
Three to get ready.
Four to go!

L

One for the money.
Two for the show.
Three to get ready.
Four to go!

BL

One for the money.
Two for the show.
Three to get ready.
Four to go!

EL

One for the money.
Two for the show.
Three to get ready.
Four to go!

NL

One for the money.
Two for the show.
Three to get ready.
Four to go!
```

If you position the cursor in the middle of the line and continue to use the forward direction, the only entity that results in a different cursor position is BL.

```
One for the money.
Two for the show.
Three to get ready.
Four to go!

BL

One for the money.
Two for the show.
Three to get ready.
Four to go!
```

5.4.5.1 The String Entity — The string entity refers to the characters *between* the cursor and the string that you type with the command. With the "move" command, EDT passes over each character between the cursor and the start of the string. You can locate strings with the "move" command because EDT moves the cursor to the position after the group of characters that is bounded by the string.

When you specify a string entity with a nokeypad command, EDT performs the operation on the text that lies *between* the cursor and the quoted string. The quoted string is not affected.

```
This example shows what happens when you
use the D (delete) command with a quoted
string as the entity.

D"the D"

the D (delete) command with a quoted
string as the entity.
```

Unlike the string entity, the string specifier, which you use with the S (substitute) and SSEL (search and select) commands, refers to the actual string characters you type. When you type S/1983/1984/, EDT finds **1983** and replaces it with **1984**. The intervening text is not affected.

5.4.5.2 The V Entity — When you use the V entity to move the cursor, you will notice that the cursor moves to the corresponding character on the next line that is in the same vertical column. Remember, however, that V is really the group of characters that includes all the characters starting with the initial cursor position, continuing to the end of that line, including the line terminator and the characters on the beginning of the next line up to the new cursor position. You can see the results best when you use the the D (delete) command.

If EDT's direction is forward, the command DV deletes all the characters to the right of the initial cursor position up to the corresponding column on the line below.

```
George Washington, President
Martha Washington, First Lady

DV

George Washington, First Lady
```

Sometimes when you use V, a line included in the entity might not be long enough to have a character in the current column position. In this case, EDT uses the end of the line as the point of reference. When using V as the "move" command, EDT keeps track of the original column position and maintains it whenever possible.

```
        March 31, 1985
        June 30, 1985
        September 30, 1985
        December 30, 1985

V

        March 31, 1985
        June 30, 1985
        September 30, 1985
        December 30, 1985

V

        March 31, 1985
        June 30, 1985
        September 30, 1985
        December 30, 1985

V

        March 31, 1985
        June 30, 1985
        September 30, 1985
        December 30, 1985
```

If there is a horizontal tab character in the line below, EDT moves to the left of the proper column to the tab character. In the following example, the text on the second line has a horizontal tab character separating the first word from the second. When you use the V entity to move the cursor from the middle of the first line to the second line, EDT shifts the cursor to the beginning of the space controlled by the horizontal tab character.

```
        The employee listings appear this way:
                name    phone number    location
        Only type the extension number, not the first two digits.

V

        The employee listings appear this way:
                name    phone number    location
        Only type the extension number, not the first two digits.

V

        The employee listings appear this way:
                name    phone number    location
        Only type the extension number, not the first two digits.
```

Notice that the cursor returned to the correct vertical column in the third line, which has no horizontal tabs.

5.5 Using Nokeypad Editing

Although people with VT100-type and VT52 terminals generally use keypad mode, it is possible to carry on your entire editing session with nokeypad editing. This section shows how to start your EDT session in nokeypad mode, move the cursor around in your text, perform various editing functions, and format text. Nokeypad commands that exist only for use in keypad key definitions are listed, but information about them appears in Chapter 7. The XLATE command, which can only be used with callable EDT on the VAX/VMS operating system, is explained in Appendix D.

The nokeypad commands can be divided into eight categories:

Moving the Cursor
- "move"

Exiting Nokeypad Mode
- EX (exit to line mode)
- EXT (extend)
- QUIT

Editing Text
- CHGC (change case)
- CHGL (change case lower)
- CHGU (change case upper)
- D (delete)
- DESEL (deactivate select)
- DLWC (default lowercase)
- DMOV (default move)
- DUPC (default uppercase)
- I (insert)
- R (replace)
- S (substitute)
- SEL (select)
- SN (substitute next)
- SSEL (search and select)
- TGSEL (toggle select)
- UNDC (undelete character)
- UNDL (undelete line)
- UNDW (undelete word)

Moving Text
- APPEND
- CUT
- KS (KED substitute)
- PASTE

Formatting Text
- FILL
- TAB
- TADJ (tab adjust)
- TC (tab compute)
- TD (tab decrement)
- TI (tab increment)

Affecting the Editing Session
- ADV (advance)
- BACK
- BELL
- CLSS (clear search string)
- REF (refresh)
- SHL (shift left)
- SHR (shift right)
- TOP

Defining Keys
- DEFK (define key)
- HELP

Special
- ASC (ASCII)
- ^ (circumflex)
- DATE
- XLATE

When you type a nokeypad command, it appears on the 23rd line of the screen. The cursor moves to the command line so that you can keep track of the commands you are typing. You can use the DELETE key to edit your command line. If you decide that you want to return the cursor to the screen, use CTRL/U to delete the entire command line and restore the cursor to the text.

5.5.1 Starting to Edit in Nokeypad Mode

When you call up EDT to edit a file, you start off in line mode. If you are using EDT to create a new file, the first thing you see is the message **Input file does not exist**, followed by the end of buffer mark ([EOB]). If you are editing an existing file, you see the first line of the MAIN buffer. The next thing that EDT displays is the line mode asterisk prompt (*) to show that you can enter line mode commands.

To use nokeypad editing, you must issue the SET NOKEYPAD line mode command. When EDT has processed this command and returned the asterisk prompt, enter the line mode CHANGE command. You are now in nokeypad mode.

This example uses EDT to create a new file.

```
☞ EDIT/EDT NEWFILE.DAT
Input file does not exist
[EOB]
*SET NOKEYPAD
*CHANGE
```

```
[EOB]
  .
  .
  .
Input file does not exist
```

If you are editing an existing file rather than using EDT to create a new one, your session might start off this way:

```
☞ EDIT/EDT LETTER12.DAT
    1       September 11, 1987
*SET NOKEYPAD
*CHANGE
```

```
September 11, 1987

Ms. Marie Stitzel
Old County Road
New Ipswich, NH  03071


Dear Ms. Stitzel:
  .
  .
  .
```

5.5.2 The Three Forms of the Line Mode CHANGE Command

The line mode CHANGE command has three forms:

```
CHANGE

CHANGE [=buffer] [range]

CHANGE ;nokeypad-command(s)
```

All three forms can be used to shift EDT to any of the three change modes (keypad, nokeypad, or hardcopy). The first form shifts to a change mode and positions the cursor on EDT's current line. If you have no text in your buffer, the cursor is at the end of buffer mark ([EOB]).

The second form allows you to include a line mode buffer and/or range specifier so that EDT positions the cursor wherever you want it. With the third form, you can issue nokeypad commands on the same line as the line mode CHANGE command. For example:

```
*CHANGE
```
Shifts EDT to a change mode.

```
*CHANGE =EXTRA
```
Shifts EDT to a change mode and moves to the buffer named EXTRA. If that buffer does not exist, EDT creates it and moves there. If the buffer already exists, EDT moves to the first line of that buffer.

```
*CHANGE ;SEL PAR FILL
```
Shifts EDT to a screen mode, creates a select range from the first paragraph, and reformats the lines in that paragraph.

5.5.3 Ending Your Nokeypad Editing Session

Before you learn how to edit a file with nokeypad commands, you need to know how to end your nokeypad editing session. Two ways are:

- Using the nokeypad EX (exit to line mode) command, followed by the line mode EXIT or QUIT command.
- Using the nokeypad QUIT command.

You are already familiar with the line mode EXIT and QUIT commands, which are described in Chapters 2 and 4.

When you use the nokeypad QUIT command, you do not save a copy of your editing work. The nokeypad QUIT command immediately ends your session and returns you to the system command level. No file is added to any directory. The same thing happens when you use the line mode QUIT command. The important difference is that with the line mode QUIT command, you can use the /SAVE qualifier to save a copy of the journal file. (See Chapters 2 and 7 for information on EDT journal files.) Nokeypad commands cannot take qualifiers.

This example ends your nokeypad editing session with the nokeypad QUIT command.

```
Sincerely yours,


James K. Polk
Director of Marketing

cc:  Ralph E. Knickerbocker
   .
   .
   .
QUIT
☞
```

The nokeypad EX (exit to line mode) command does not end your editing session. It merely ends your nokeypad work and shifts EDT to line mode. As soon as you type EX in nokeypad, you see the line mode asterisk prompt (*) on your screen.

```
Sincerely yours,


James K. Polk
Director of Marketing

cc:  Ralph E. Knickerbocker
   .
   .
   .
EX
*
```

Once you are in line mode, you can use either the line mode EXIT or QUIT command to end your EDT session.

```
cc:  Ralph E. Knickerbocker

*EXIT
DISK$USER:[POLK]LETTER1.RNO;2 43 lines
☞
```

5.5.4 Using Line Mode Commands in Nokeypad Mode

You can issue the line mode EXIT command directly from nokeypad mode using the nokeypad EXT (extend) command. Simply type EXT followed by the line mode command while in nokeypad mode.

Two things to remember about EXT are:

- EXT cannot take a count specifier.
- You can use spaces within an EXT command as long as they are valid in the line mode command.

These examples show how to use EXT to end your nokeypad editing session:

```
EXT EXIT
```
Ends your EDT session and saves a copy of the MAIN buffer text.

```
EXT EXIT /SAVE
```
Ends your EDT session and saves a copy of the MAIN buffer text and the journal file.

```
EXT EXIT STOCKMEMO.RNO
```
Ends your EDT session and saves a copy of the MAIN buffer text in the external file STOCKMEMO.RNO.

```
EXT QUIT /SAVE
```
Ends your EDT session and saves only a copy of the journal file. The MAIN buffer text is discarded.

You can use EXT with most other line mode commands. For example, EXT with the SHOW BUFFER command tells you which buffers currently exist in your editing session. Using EXT with the FIND command, you can have EDT move to another buffer. You can use EXT with WRITE to copy part or all of a buffer to an external file. The commands might look like this:

```
EXT SHOW BUFFER
 ADDR2   5        lines
 ADDR1   6        lines
=MAIN    123      lines
 PASTE   9        lines
Press return to continue
```
Displays all the buffers that currently exist for your editing session. In order to resume editing in nokeypad mode, press the RETURN key.

```
EXT FIND =ADDR2
```
Moves EDT to the first line of the buffer named ADDR2.

```
EXT WRITE LETTERA.RNO =LETTERA
```
Puts a copy of the text from LETTERA in a file named LETTERA.RNO. Does not end your EDT session; does not move to buffer LETTERA; has no effect on the text in either the current buffer or LETTERA. EDT prints a message indicating how many lines of text were copied to the external file.

```
EXT INCLUDE ADDRESS3.DAT =ADDR3
```

Puts a copy of the file ADDRESS3.DAT in the buffer ADDR3. Moves to ADDR3.

```
EXT DELETE REST
```

Deletes all the lines in the current buffer from the current line to the end of the buffer.

5.5.5 Getting Online Help about Nokeypad Editing

You can get information on nokeypad editing by using the line mode HELP command. Nokeypad information is contained under the HELP topic CHANGE. HELP CHANGE has five subtopics:

```
ENTITIES
HARDCOPY
KEYPAD
SCREEN
SUBCOMMANDS
```

The ENTITIES subtopic contains a list of the nokeypad entity subtopics: character, word, line, range, sentence, page, paragraph, select, vertical, and string. When you want online information about one of these entities, for instance sentence, include the entity name as the last HELP command subtopic:

```
EXT HELP CHANGE ENTITIES SENTENCE
```

The HARDCOPY, KEYPAD, and SCREEN subtopics provide brief descriptions of hardcopy change mode, keypad mode, and screen editing. They do not have further subtopics.

Information on nokeypad commands is contained in the SUBCOMMANDS subtopic. For a complete list of the subtopics covered, type:

```
EXT HELP CHANGE SUBCOMMANDS
```

For help on a particular command, for instance PASTE, type:

```
EXT HELP CHANGE SUBCOMMANDS PASTE
```

If you are not sure of the exact subtopic you want information on, you might find it easier to switch to line mode so you can look around in the HELP file without having to type EXT for each line mode HELP command. Use the EX command to shift to line mode. (See Chapter 4 for more information on using the line mode HELP command.)

The nokeypad HELP command does not give information on nokeypad editing. It is used for key definitions only. (See Chapter 7 for information on using nokeypad commands to define keys for keypad editing.)

5.6 Editing Text in Nokeypad Mode

Editing in nokeypad mode involves the same operations as in keypad and line mode – inserting and deleting text, moving around in a buffer, and performing substitutions. In addition, nokeypad mode has commands that enable you to move and copy text.

5.6.1 Inserting Text – I

If you are using EDT to create a file, the first command you need is the I (insert) command. This command allows you to put some text in your buffer. To put EDT into the insert state, type the letter I. The I appears at the bottom of the screen on the command line.

After you type the I command and press RETURN, the cursor moves back to its current position on the screen. The characters you type now are inserted to the left of the cursor. You can type anything from a few characters to pages of text without leaving the insert state. If there is some text in the buffer, EDT adds the new text to it. If you start with an empty buffer, EDT places the new text at the top of the buffer.

When you want to start another line of text, simply press the RETURN key to move the cursor to the next line. EDT is still in the insert state. You can verify this by checking to see if the I command is still displayed on the command line. The I remains on the screen until you exit from the insert state.

If you want to correct a mistake or change a word, you can use the DELETE key to delete characters to the left of the cursor. If your mistake was on the previous line, you can continue to press DELETE until you reach the text you want to change. Then retype the text you had to delete. The DELETE key is the only way you can edit text while still in the insert state. Generally, it is easier to finish inserting the text and then use other nokeypad editing commands to make changes or corrections.

When you finish inserting the new text, press CTRL/Z to exit from the insert state. The I command disappears from the command line and you are ready to issue more nokeypad commands.

The following example shows the CTRL/Z key sequence at the end of the last line. When you press CTRL/Z to leave the insert state, EDT does not display any characters on the screen. Notice the I command at the bottom of the screen. This disappears as soon as EDT processes CTRL/Z.

```
I(RETURN)

        DATE:       April 15, 1984
        TO:         All Developers
        FROM:       Mark Harding
        SUBJECT:    Travel Arrangements (CTRL/Z)
          .
          .
          .
        I
```

The I command has two forms:

```
I (RETURN) text (CTRL/Z)

Itext^Z
```

The first one shifts EDT to the insert state so you can enter more than one line of text at a time. The second form is limited to entering text on a single line. You can use the latter form to insert a few words to the left of the cursor. Simply type the I command and follow it immediately by the text you want inserted. If you put a space between the I and the text, EDT inserts a space in your text. Follow the text immediately by pressing CTRL/Z or typing the circumflex (^) and the letter Z. Again, if you separate the text from ^Z with a space, EDT will insert that space in your text.

The next example uses the single line I command to insert the word **Wednesday,**(SP) before **August**. EDT inserts the text to the left of the cursor.

```
     on (A)ugust 31, 1984.

IWednesday,▵^Z

     on Wednesday, (A)ugust 31, 1984.
```

5.6.2 Moving the Cursor — The "move" Command

The "move" command moves the cursor to a new location in the current buffer. Like the line mode <null> command, it has no command word or abbreviation. The syntax is simply the entity specifier with an optional sign or count specifier.

```
[+|-][count]entity
```

The entity you specify for the "move" command determines how far EDT moves the cursor. The count specifier enables you to move the cursor more than one entity at a time. For example, if EDT's direction is forward, 3W moves the cursor to the beginning of the fourth word to the right of the current cursor position.

The sign specifier reverses the current direction only for that "move" command. If the current direction is forward (the default) EDT moves from the current cursor position toward the bottom of the buffer. Use the minus sign to go back one or more entities. If the current direction is BACK, use the plus sign to move forward one or more entities. Remember that entities beginning with B or E are not affected by either sign or editing direction, except for EL.

You can use the following paragraph to try out different "move" commands. Begin with the cursor at the top left character of the text.

```
     (T)o process data, you must know how to give the computer exact
instructions (commands).  DATATRIEVE is a system of commands
that you use to instruct the computer to perform various
operations on your established data.  DATATRIEVE commands are
familiar English words.  The command sequences have sentence-
like structures.
```

First move the cursor four words to the right.

```
4W
    To process data, you must know how to give the computer exact
```

Now move the cursor to the beginning of the next sentence.

```
SEN
    instructions (commands).  DATATRIEVE is a system of commands
```

Remember that although the space is a word boundary character, EDT considers it to be part of the word, not a separate word. Thus, when you use EW as a "move" command, EDT moves the cursor to the beginning of the next word.

```
EW
    instructions (commands).  DATATRIEVE is a system of commands
```

Next, move the cursor four characters to the left.

```
-4C
    instructions (commands).  DATATRIEVE is a system of commands
```

Now move the cursor down three lines.

```
3L
    familiar English words.  The command sequences have sentence-
```

To move the cursor to the third word after the end of the sentence, type:

```
SEN2W
    familiar English words.  The command sequences have sentence-
```

Next move the cursor back to the end of the fourth line above the current line.

```
-4EL
    To process data, you must know how to give the computer exact
```

Use the string entity to move the cursor to the next occurrence of **data**.

```
"data"
    instructions (commands).  DATATRIEVE is a system of commands
```

You can combine the string entity with another entity. This example shows two "move" commands on the same line:

```
"fami" 2W

    familiar English words.  The command sequences have sentence-
```

5.6.3 Deleting Text – D

Once you know how to move the cursor and specify entities, you can learn how to use the D (delete) command.

The syntax for the D command is:

```
[+|-][count]D[+|-][count]entity
```

The D command deletes the specified entity. There is no default entity specifier. If you type D with no entity, EDT prints the message **Invalid entity**. If the cursor is on a character and you type DC, that character is deleted. If the cursor is anywhere within a word and you type DW, the entire word is deleted. If the cursor is anywhere within a line and you type DL, the entire line is deleted.

When you use the D command with a B entity prefix (for example, BL or BSEN), EDT deletes all the characters to the left of the cursor up to the entity boundary. For E entities (for example, ER or ESEN), D deletes all the characters (starting with the cursor character) to the right up to the entity boundary. (Space boundaries are deleted along with the entity if EDT considers the space to be part of the entity.) If you use D with the string entity, EDT deletes all the characters between the cursor character and the string; the string itself remains in your text.

The next examples use the D command to delete various entities. Several examples use both the "move" command and the D command. When you combine these two commands you can separate them with a space, but do not separate the entity you want to delete from its D command. Using the following list of names, you can see how the D command works with different entity specifiers:

```
Sonja Garvy
Leslie I. Hillis
Herbert Kline
Alexander H. Morris
Milton Newman
```

You can use DL to delete the entire first line even though the cursor is in the middle of the line.

```
DL

    Leslie I. Hillis
    Herbert Kline
    Alexander H. Morris
    Milton Newman
```

Now move the cursor over two characters and delete the middle three letters in the name.

```
2C D3C

    Le I. Hillis
    Herbert Kline
    Alexander H. Morris
    Milton Newman
```

Next, move the cursor to the middle initial and use DW to delete both the **I** and the period.

```
W DW

    Lee Hillis
    Herbert Kline
    Alexander H. Morris
    Milton Newman
```

You can delete the middle initial on the third line by using the "move" command to position the cursor on the **H** and then using DW to delete both that letter and the period.

```
"H." DW

    Lee Hillis
    Herbert Kline
    Alexander Morris
    Milton Newman
```

Using DNL, you can delete from the cursor to the end of the line including the line terminator.

```
DNL

    Lee Hillis
    Herbert Kline
    Alexander Milton Newman
```

Notice what happens in the last two examples. In the example where you deleted the string **H.**, you first had to move the cursor to that string using the "move" command. Then you deleted the new current word. If you type D"H.", all the characters from **Hillis** up to **H.** will be deleted; the string **H.** will still be in the text because the string entity refers to the characters between the original cursor position and the first string character. In the DNL example, all the characters starting with the cursor and including the line terminator were deleted up to the first character on the next line.

Now take the list as it existed in the last example and restore it to its original state using I (insert) and "move". The next examples use the I command in its single line version as well as in the multiline format.

Start reconstructing the list with the cursor where you left off.

```
    Lee Hillis
    Herbert Kline
    Alexander Milton Newman

3BL
2C Isli^Z
W II.▵^Z

    Leslie I. Hillis
    Herbert Kline
    Alexander Milton Newman

BL I(RETURN)

Sonja Garvy(RETURN)
(CTRL/Z)

    Sonja Garvy
    Leslie I. Hillis
    Herbert Kline
    Alexander Milton Newman

"Milt"I(RETURN)

H. Morris(RETURN)
(CTRL/Z)

    Sonja Garvy
    Leslie I. Hillis
    Herbert Kline
    Alexander H. Morris
    Milton Newman
```

5.6.4 The Undelete Commands — UNDC, UNDW, UNDL

When you use the D command to delete a character, word, or line, EDT stores the deleted entity in a special buffer. Only the most recently deleted entity is stored. The last character you delete is stored in the delete character buffer, the last word you delete is stored in the delete word buffer, and the last line you delete is stored in the delete line buffer. These buffers are not accessible and do not appear in the SHOW BUFFER list.

The nokeypad undelete commands allow you to insert the contents of these buffers into your text at any time. The three commands are:

```
[count]UNDC

[count]UNDW

[count]UNDL
```

UNDC inserts the contents of the delete character buffer, UNDW the delete word buffer, and UNDL the delete line buffer. Each command takes the count specifier, enabling you to repeat the insert in the same location.

Use DC or the DELETE key to put a character in the delete character buffer. DW, DBW, and DEW put text in the delete word buffer. DL, DBL, DEL, and DNL all move text to the delete line buffer.

Suppose you want to type a line of 50 underscores. Using I, DC, and UNDC, you can have EDT create the line for you.

First use the I command to enter one underscore in your buffer. Next, use the D command to delete the underscore and put it in the delete character buffer. Since the cursor is on the character just after the one you inserted, you must use D–C (or –DC) to delete the underscore. Finally, use the UNDC command with a count of 50 to create your line.

```
I_^Z

D-C

50UNDC

     ----------------------------------------------------
```

Because nokeypad mode allows multicommand lines, you can type all three commands at the same time.

```
I_^Z D-C 50UNDC
```

The UNDW and UNDL commands work in a similar fashion. However, because there are more characters involved, you must be aware of spaces in the delete word buffer and line terminators in the delete line buffer. Suppose you want to store several characters in the delete word buffer, but do not want to include a space. Insert the characters at the end of a line, move the cursor back to the first character in the group, and then issue the delete word command. EDT deletes all characters up to the line terminator boundary so no space is put in the delete word butter. You can use this method to create a zigzag line by putting a slash and backslash in the buffer.

```
     Please fill out the form below.

I/\^Z

     Please fill out the form below. /\

D-W

     Please fill out the form below.

2L

25UNDW

     Please fill out the form below.

     /\/\/\/
```

If you delete a line portion without including a line terminator, you can store several words in the delete line buffer and use UNDL to insert these words anywhere in your text.

For example, you might need to insert a date or place several times in a letter or memo. Type the date or place at the end of a line and use DEL to put the piece of text in the delete line buffer. Then, move the cursor to a location where you want to insert the date or place and issue the UNDL command.

```
     Senator Howard Baker,□Washington, DC  20510
DEL UNDL

     Senator Howard Baker,□Washington, DC  20510
"Robert Byrd" EL UNDL

     Senator Robert Byrd,□Washington, DC  20510
```

5.6.5 Combining the Delete and Insert Functions — R

The R (replace) command is a combination of the D (delete) and I (insert) commands. EDT first deletes the entity you specify with R and then shifts to the insert state. After you finish typing the new text, you press CTRL/Z to complete the insert. The syntax for R is:

```
[+|-][count]R[+|-][count]entity
```

There is no way to type your insert text in the R command line as you can with the I command or to add another nokeypad command after R. When you finish typing the R command, press RETURN to return the cursor to the location in the buffer where you want to insert some new text. Then type your text and press CTRL/Z to complete the operation.

```
     Today is April first.
RW(RETURN)

     Today is April Fool's Day.□
(CTRL/Z)
```

5.6.6 Performing Substitutions – S, SN

The S (substitute) command replaces strings in your text. The string specifier used with the S command differs from the string entity. When you use the entity string, EDT focuses on the text between the cursor and the string. When you use the string specifier, EDT considers the string itself as the text to be acted on.

The syntax for S is:

```
[+|-][count]S[/[string-1]/string-2/]
```

String-1 is always the search string. EDT searches the current buffer in the current EDT direction until it finds the next occurrence of string-1. Then EDT deletes string-1 and replaces it with string-2, the substitute string.

Whenever you supply a string-1 with your S command, EDT overwrites the contents of the search buffer with that string, just as it does when you use the string entity.

If the cursor is currently located in the middle of a string that matches string-1, EDT makes the substitution for that string.

The sign determines whether the search for string-1 occurs in the forward (+) or backward (–) direction. The sign overrides EDT's current direction.

The count specifier allows you to perform as many substitutions as you need. When you want to perform substitutions on every string starting with the current cursor position and moving either to the end or beginning of the buffer, simply use a large number for the count specifier. EDT performs as many substitutions as possible. When it cannot find another string-1 in the remaining portion of the buffer, it prints the message **String was not found** and positions itself next to the last string-2 which it inserted in the text.

Here are some examples of the S command. You can see how each line changes.

```
     August 14, 1983

     Mr. Robert T. Jamison
     3973 Great Falls Street
     Washington, DC  20033


S/14/25/

     August 25, 1983

S/mison/sperson/

     Mr. Robert T. Jasperson

S/street/Road/

     3973 Great Falls Road

2S/3/2/

     Washington, DC 20022

-S/83/84/

     August 25, 1984

     Mr. Robert T. Jasperson
     3973 Great Falls Road
     Washington, DC  20022
```

Notice that when you change **Street** to **Road**, you do not have to type an uppercase **S** for the search string. EDT generally disregards case and diacritical marks when looking for strings in your text. (For more information on how EDT handles searches, see the section on SET SEARCH in Chapter 6.) However, you must always type the substitute string exactly as you want it to appear in your text.

When you use the minus sign to reverse direction in changing 1983 to 1984, EDT leaves the cursor on the **9** after processing the command. EDT's direction is still forward, so you do not need to give the ADV command before issuing your next editing command.

You can omit string-1 or string-2 from the S command under certain circumstances. If you omit string-1, EDT always uses the current search string. If the search buffer is empty, EDT simply inserts string-2 into the text. If you omit string-2, EDT always deletes string-1 and inserts nothing. The following samples show what happens when you omit various elements:

```
This is file A.      S/file/buffer/      This is buffer A.
This is file B.      S///                This is  B.
This is file C.      S/file//            This is  C.
This is file D.      S//buffer/          This is buffer D.
This is file E.      S                   This is  E.
```

You can use the SN (substitute next) command only after you have established both the search string and substitute string. Although SN can take a sign or count specifier, it uses no string specifiers. The syntax is:

```
[+|-][count]SN
```

Since there are no string specifiers with this command, the string that you want to delete must be in the search buffer and the replacement string must be in the substitute buffer. The easiest way to do this is to issue an S command that uses the strings you want just before giving the SN command. Remember that the line mode SUBSTITUTE and SUBSTITUTE NEXT commands affect the contents of the search and substitute buffers. And, any command or function in any mode that uses the search buffer affects the text that is stored there.

For example, if you use a string entity between your S and SN commands, that string entity becomes the contents of the search buffer, replacing the string-1 you used with the S command.

You can use SN after testing a substitution with the S command. Once you have the search and substitute strings set correctly, use the SN command for the remaining substitutions. The sign specifier allows you to make substitutions in either direction; the count specifier enables you to perform two or more substitutions with the same SN command. For example:

```
January 3, 1983
January 11, 1983
January 25, 1983
January 31, 1983

S|january␣|1/|

1/3, 1983
January 11, 1983
January 25, 1983
January 31, 1983

3SN

1/3, 1983
1/11, 1983
1/25, 1983
1/31, 1983
```

```
S|, 1983|/83|

      1/3, 1983
      1/11, 1983
      1/25, 1983
      1/31/83

-3SN

      1/3/83
      1/11/83
      1/25/83
      1/31/83
```

5.6.7 Commands That Change Case — CHGC, CHGL, CHGU, DLWC, DMOV, DUPC

Two groups of nokeypad commands affect the case of letters. The first group – CHGC, CHGL, and CHGU – change the case of letters in the entities you specify. The second group – DLWC, DMOV, and DUPC – affect the way EDT functions as it moves through the text being edited. The syntax for the entity changing commands is:

```
[+|-][count]CHGC[+|-][count]entity

[+|-][count]CHGL[+|-][count]entity

[+|-][count]CHGU[+|-][count]entity
```

The only difference in the syntax for these three commands is the last letter of each command: C, L, or U. The CHGC (change case) command changes the case of every letter in the entity. All uppercase letters become lowercase; all lowercase letters become uppercase. The CHGL (change case lower) command changes every uppercase letter in the entity to lowercase; lowercase letters remain lowercase. In contrast, the CHGU (change case upper) command changes every lowercase letter in the entity to uppercase; uppercase letters remain uppercase.

The sign specifier enables you to reverse the direction of the command without resetting EDT's direction. The count specifier determines how many times the command is repeated or how many entities the command affects. The entity specifier determines which entity the command affects.

This example shows how the three commands affect the same line:

```
      Romeo and Juliet

CHGCL

      rOMEO AND jULIET

CHGLL

      romeo and juliet

CHGUL

      ROMEO AND JULIET
```

The three other commands – DLWC, DMOV, DUPC – affect the default way that EDT handles case for both screen modes as well as hardcopy change mode. These commands do not take any specifiers because they directly affect the case of every letter EDT passes over with a move operation.

DMOV (default move) is the normal state for EDT. Under DMOV, the case of letters is unaffected as EDT moves about in a buffer.

DLWC (default lowercase) sets your editing session to lowercase. Whenever you move the cursor, all uppercase letters passed over by the cursor are changed to lowercase.

DUPC (default uppercase) sets your editing session to uppercase. EDT changes all lowercase letters that the cursor passes over to uppercase.

Suppose after you type a notice, you decide that it will look better with all the letters on certain lines in uppercase. You can make those changes by using the DUPC command and the appropriate "move" commands.

```
                Important Meeting

              Marketing Department

        All Sales Representatives Must Attend

                  April 26, 9:30 a.m.
                  Rodgers Auditorium

                  Robert R. Robinson
                     Guest Speaker

DUPC "april"

DMOV "robert"

DUPC L DMOV

                IMPORTANT MEETING

              MARKETING DEPARTMENT

        ALL SALES REPRESENTATIVES MUST ATTEND

                  April 26, 9:30 a.m.
                  Rodgers Auditorium

                  ROBERT R. ROBINSON
                     Guest Speaker
```

Notice that when you used the string specifier for the "move" command, EDT changed the case of all letters between the current cursor position and the word **April**. In changing the case of letters in the speaker's name, you could have used CHGUL instead of DUPC L DMOV.

5.6.8 Using the Select Range — SEL, DESEL, TGSEL, SSEL

The select commands create and affect the select range entity – SR. You can use select ranges with editing and formatting commands as well as commands that move text. The syntax for the four select commands is:

```
SEL

DESEL

TGSEL

[+|-]SSEL[+|-]"string"
```

5.6.8.1 SEL — You use the SEL (select) command to mark one end of the select range. Mark the other end by using "move" commands or the arrow keys to move the cursor to that point. You can move the cursor either to the right or to the left of the position that was fixed with the SEL command. On VT100-type terminals, the selected text is displayed in reverse video. On VT52 terminals you must remember the starting point.

Once you have established the select range, you can issue a command to delete, replace, change case, cut, or append the text that has been set off. Use SR as the entity specifier.

```
We are planning a farewell party for Janice
Fredericks.  Even if you cannot attend, you
can still contribute to the gift.  Contact
Mary Evans for information on the date, time,
and place.
```

```
"even" SEL SEN
```

```
We are planning a farewell party for Janice
Fredericks.  Even if you cannot attend, you
can still contribute to the gift.  Contact
Mary Evans for information on the date, time,
and place.
```

```
DSR
```

```
We are planning a farewell party for Janice
Fredericks.  Contact
Mary Evans for information on the date, time,
and place.
```

The select range entity can contain one character (although this would be rare) or hundreds of lines. Until you issue a command that uses SR as the entity, EDT continues to maintain the select range in the active state. Thus, you can issue the SEL command and then take your time about moving the cursor to the other end of the range. You can issue any number of commands to move the cursor – first a few lines, then a few words, then maybe a character or two, perhaps adding a sentence or even a paragraph along the way. Once you are sure you have included all the characters you want, give your command with SR as the entity.

If you find that you included too much material in your select range, you can move the cursor in the opposite direction to eliminate the end portion of the range. For example, suppose you issue the SEL command to set one end of your select range and then type 6L to have EDT include six lines in that range. When you look at the new cursor position, you realize that you only want five lines in the range. Simply type –L to have EDT "subtract" the last line from the active select range. Now when you issue the command, it will affect only five lines.

You can use select ranges with line mode commands such as WRITE or COPY. The line mode range specifier SELECT refers to a screen mode select range. Remember that when you use a select range with a line mode command, the select range can include only whole lines.

These examples show how you might use a select range with a line mode command directly from nokeypad mode.

```
EXT WRITE OUTFILE.DAT SELECT
```

Puts a copy of the select range in the file OUTFILE.DAT.

```
EXT COPY SELECT TO =EXTRA END
```

Puts a copy of the select range at the end of the buffer named EXTRA. EDT moves to that buffer.

5.6.8.2 DESEL – The DESEL (deactivate select) command cancels an active select range. There are no specifiers with this command; its only function is to negate the SEL command.

5.6.8.3 TGSEL – TGSEL (toggle select) toggles between SEL and DESEL. If a select range is active, TGSEL performs the DESEL function. If no select range is active, TGSEL performs the SEL function. As with SEL and DESEL, this command takes no specifiers.

5.6.8.4 SSEL – The SSEL (search and select) command takes two specifiers: sign and string. The string specifier works like the string-1 specifier for the S command, not like the string entity. SSEL searches for the string that is supplied with the command and creates a select range containing that string. The syntax is:

```
[+|-]SSEL[+|-]"string"
```

You can use SSEL to find a string and delete it. Remember that if you use the D (delete) command with a string entity, EDT deletes everything from the current cursor position up to the specified string. Using SSEL and DSR, you delete the string itself.

```
I will definitely attend the meeting.
SSEL"definitely "
DSR

I will attend the meeting.
```

The string used with SSEL becomes the current contents of the search buffer. When you type the string specifier with the SSEL command, be sure to enclose the characters in quotation marks, either single (') or double ("). If the string you want to select is already in the search buffer, all you need type is the quotation marks.

```
SSEL""
```

5.6.9 Moving and Copying Text — CUT, PASTE, APPEND, KS

Nokeypad has three commands for copying and moving text: CUT, PASTE, and APPEND. These three commands can take a buffer specifier so that you can move text to and from different buffers. The KS command can only be used in combination with PASTE; it simply modifies the cursor position after the PASTE command has been processed.

Moving and copying text in nokeypad mode is similar to those operations in keypad editing. You must move the text from the current buffer into another buffer and then copy it from that other buffer to the new location.

Moving text involves three steps:

1. Use the CUT command to delete the text from its current location.
2. Move the cursor to the location where you want the text.
3. Use the PASTE command to insert the text at the new location.

Copying text involves four steps:

1. Use the CUT command to delete the text from its current location.
2. Use the PASTE command to restore the text to its original location.
3. Now move the cursor to the location where you want the text.
4. Use the PASTE command to insert the text at the new location.

5.6.9.1 CUT — You use the CUT command to remove the text from the current buffer and move it to another buffer. The syntax for CUT is:

```
[+|-][count]CUT[+|-][count]entity[=buffer]
```

You must supply the entity specifier with the CUT command. The other specifiers are optional. The sign specifier indicates which direction to make the cut. The count specifier tells EDT how many entities to delete or how many times to perform the CUT operation.

When you do not include the buffer specifier, EDT moves the text to the PASTE buffer. Only the text is moved; EDT remains in the current buffer.

The buffer specifier allows you to use a buffer other than PASTE to store the text. If you specify a buffer name that does not exist, EDT creates a buffer with that name, moves the text there, but still remains in the current buffer.

You must precede the buffer name with an equal sign signal (=). Do not separate the equal sign and buffer name from the rest of the nokeypad command.

Each time you use CUT to move text to a buffer, EDT overwrites the contents of that buffer. For example, if you use CUT to move three lines to the STORE buffer and later use CUT to move four words to the same buffer, the STORE buffer will then have only the four words that were moved with the second CUT command.

You can use the CUT command to simply delete text from the buffer you are working in. But generally you use CUT in combination with the PASTE command. For this reason, examples of CUT appear in the next section on the PASTE command.

5.6.9.2 PASTE — The PASTE command inserts a copy of the text in the specified buffer to the left of the cursor in the current buffer. The syntax for PASTE is:

```
[count]PASTE[=buffer]
```

Notice that there is no entity specifier with the PASTE command. You can only copy the entire contents of the buffer. If you do not need the entire text, use a line mode command to enter that buffer, delete the unwanted text, and then return to your original buffer (again using a line mode command). Now you are ready to give the PASTE command.

The buffer specifier enables you to copy text from a buffer other than the PASTE buffer. If you omit the buffer specifier, EDT copies the text that is in the PASTE buffer. Suppose you use a buffer specifier with the CUT command. If you omit the buffer specifier with the next PASTE command, EDT inserts whatever text is currently in the PASTE buffer, not the text that you deleted with the previous CUT command. If you supply the name of a buffer that does not exist with your PASTE command, EDT creates an empty buffer with that name. However, there is no change to the text in the current buffer.

You must precede the buffer name with the equal sign (=) signal. As with CUT, you cannot separate the equal sign and buffer name from the rest of the nokeypad command.

The count specifier allows you to copy the contents of the buffer several times at the same location.

Just as you can use the CUT command alone to delete text from your buffer, you can use PASTE to insert the contents of any buffer into your current buffer at any time. However, most of the time you will be using the two commands in succession to move or copy text from one place to another.

With the delete commands and their corresponding undelete commands, you can only move or copy characters, words, or lines. The CUT/PASTE command sequence allows you to use any entity or combination of entities to rearrange paragraphs, sentences, groups of several lines, and so forth.

The following example alphabetizes a list of names and addresses. The first command uses a select range to move the address entry. The remaining commands use the line entity, which works equally well for lists.

```
Irving Randolph
342 Lowell Street
Andover, MA  01812

John Franklin
273 Chester Street
West Somerville, MA  02144

Leonard Michaels
211 North Main Street
Concord, NH  03301

Neville Jackson
Depot Road
Reading, VT  05062

Judith Allan
34 Memorial Road
Providence, RI  02906
```

When you look down the list, you notice that the last entry should be the first. Move the cursor to the **J** in **Judith**. Then give the SEL command and a "move" command of 4L. Be sure to include the blank line following the city/state line.

```
"jud" SEL 4L

CUTSR=ADDR
```

Next, using the minus sign to reverse EDT's direction, move the cursor back to the first character at the top of the list and issue the PASTE command.

```
-"Irving" PASTE=ADDR
```

```
Judith Allan
34 Memorial Road
Providence, RI  02906

Irving Randolph
342 Lowell Street
Andover, MA  01812

John Franklin
273 Chester Street
West Somerville, MA  02144

Leonard Michaels
211 North Main Street
Concord, NH  03301

Neville Jackson
Depot Road
Reading, VT  05062
```

The cursor is now located on the name and address that should be last in the list, so you might as well move this entry now. This time, use 4L as the specifier with the CUT command – CUT4L. Next, move the cursor to the end of the buffer and use PASTE to insert the text.

```
CUT4L=ADDR

ER PASTE=ADDR
```

```
Judith Allan
34 Memorial Road
Providence, RI  02906

John Franklin
273 Chester Street
West Somerville, MA  02144

Leonard Michaels
211 North Main Street
Concord, NH  03301
```

```
Neville Jackson
Depot Road
Reading, VT  05062

Irving Randolph
342 Lowell Street
Andover, MA  01812

[EOB]
```

By moving the Neville Jackson entry, you can complete the job. Put the cursor on the **N** in **Neville** and then use CUT to move the text to the ADDR buffer. Then place the cursor on the **L** in **Leonard**, and copy the text from the ADDR buffer to the new location.

```
"Neville" CUT4L=ADDR

-"Leonard" PASTE=ADDR

     Judith Allan
     34 Memorial Road
     Providence, RI  02906

     John Franklin
     273 Chester Street
     West Somerville, MA  02144

     Neville Jackson
     Depot Road
     Reading, VT  05062

     Leonard Michaels
     211 North Main Street
     Concord, NH  03301

     Irving Randolph
     342 Lowell Street
     Andover, MA  01812
```

Copying text from one location to another is similar to moving the text.

```
     DATATRIEVE is an interactive tool for inquiry,
     update, and maintenance of information stored
     in databases.  This book teaches you how to
     write commands and statements to perform
     the basic tasks of interactive data manipulation.

CUTW PASTE
"commands" PASTE

     DATATRIEVE is an interactive tool for inquiry,
     update, and maintenance of information stored
     in databases.  This book teaches you how to
     write DATATRIEVE Commands and statements to perform
     the basic tasks of interactive data manipulation.
```

5.6.9.3 APPEND — The APPEND command performs the same type of function as CUT, but does not overwrite the contents of the buffer to which the text is moved. If you use APPEND with an empty buffer, it has the same effect as CUT. However, when you want to *add* text to the contents of a buffer, use APPEND. APPEND enables you to move text from two or three different locations and put the entire combination in a fourth. Remember to use the same buffer specifier with each APPEND command.

The APPEND syntax is:

```
[+|-][count]APPEND[+|-][count]entity[=buffer]
```

Using the sign specifier, you can have EDT process the command in the direction that is not EDT's current direction. The count specifier enables you to specify more than one entity (for example, APPEND3PAR). By using the buffer specifier, you can have EDT put the text in a buffer other than the PASTE buffer.

You must precede the buffer name with the equal sign (=) signal. As with CUT and PASTE, you cannot separate the equal sign and buffer name from the rest of the nokeypad command.

Using the APPEND command with CUT and PASTE, You can alphabetize the same list of names and addresses. Start by moving the cursor to the entry you want first in your list.

```
Irving Randolph
342 Lowell Street
Andover, MA  01812

John Franklin
273 Chester Street
West Somerville, MA  02144

Leonard Michaels
211 North Main Street
Concord, NH  03301

Neville Jackson
Depot Road
Reading, VT  05062

Judith Allan
34 Memorial Road
Providence, RI  02906
```

```
"Judith" CUT4L
```

With the first entry now at the top of the PASTE buffer, scan the remaining names to see which entry is second. After moving the cursor to the **J** in **John**, use APPEND to add that entry to the end of the text currently in the PASTE buffer.

```
-"John" APPEND4L
```

Next, move to the **N** in **Neville** and add that entry to the PASTE buffer. The last addition you need is the Leonard Michaels entry.

```
"Neville" APPEND4L

-"Leonard" APPEND4L
```

The final operation is to move the cursor to the **I** in **Irving** and use PASTE to insert the contents of the PASTE buffer above the cursor line.

```
-"Irving" PASTE

     Judith Allan
     34 Memorial Road
     Providence, RI  02906

     John Franklin
     273 Chester Street
     West Somerville, MA  02144

     Neville Jackson
     Depot Road
     Reading, VT  05062

     Leonard Michaels
     211 North Main Street
     Concord, NH  03301

     Irving Randolph
     342 Lowell Street
     Andover, MA  01812
```

5.6.9.4 KS — The KS (KED substitute) command affects the cursor position after EDT processes a PASTE command. It is only used after PASTE and must follow that command directly. (**CUTSR = DELETE PASTEKS""**. is the preset definition for the keypad SUBS function.)

When PASTE is used alone, the cursor is always positioned on the character to the right of the inserted text, regardless of EDT's current direction. With EDT in the forward direction, PASTEKS puts the cursor on the last character of the inserted text. With EDT in the backward direction, PASTEKS puts the cursor on the first character of the inserted text. The cursor positions that result from using the KS command make it possible for a search function to find the text immediately to the left or right of the inserted material.

Suppose you had used CUT to move a list of three Canadian cities. The next examples show the cursor position after the PASTE command is used. In the first example, PASTE is used by itself, with EDT in the forward direction. The cursor is also in the same position when PASTE is used alone with EDT in the backward direction. When PASTE is used with KS, however, the cursor position changes. In the forward direction, the cursor is located on the last character of the inserted text – the line terminator – after **Toronto**. In the backward direction, the cursor is on the first character of the inserted text, the **M** in **Montreal**.

```
PASTE

     Montreal
     Quebec City
     Toronto
     

BACK
PASTE

     Montreal
     Quebec City
     Toronto
     
```

```
ADV
PASTEKS

     Montreal
     Quebec City
     Toronto

BACK
PASTEKS

     Montreal
     Quebec City
     Toronto
```

5.7 Commands That Affect Your Editing Session

There are several nokeypad commands that affect only your EDT session, rather than the text you are editing. The commands include those that determine EDT's direction: ADV and BACK, and commands that affect the portion of the buffer visible on the screen: REF, SHL, SHR, and TOP. In addition, there is the CLSS command, which clears the search string, and the BELL command, which sounds the terminal bell.

5.7.1 Commands That Change EDT's Direction — ADV, BACK

You have already seen how ADV and BACK work in some of the examples in this chapter. ADV sets EDT's direction to forward; BACK sets it to backward.

The direction set by ADV or BACK is maintained by EDT regardless of which screen mode you are in. When you shift from one screen mode to the other, EDT maintains the current direction until you issue a new direction function.

Suppose you are working in keypad mode and press the BACKUP key. Next you shift to line mode and perform a few line mode operations. Line mode is not affected by EDT's direction, so you might forget that you set EDT to backward. If you shift to nokeypad mode, you find that the current direction is still backward.

Whenever you are uncertain about EDT's current direction, you can give the ADV or BACK command to make sure that EDT is going in the direction you want. The syntax for ADV and BACK is the command itself:

```
ADV

BACK
```

These commands can be joined with other EDT commands to reset the direction at the beginning, middle, or end of a nokeypad command line.

```
BACK "83"
```
Finds the string **83** by moving backward toward the beginning of the buffer.

```
I.mem^Z BACK ";" +C +DEL
```
Inserts the text **.mem**. Then changes to the backward direction to find the string ; on the previous line. Deletes to the end of that line. The plus signs process the C and DEL commands in the forward direction, but EDT's direction is still backward. (Remember that EL depends on EDT's current direction.)

```
BACK "Cairo" DPAR ADV
```

The current direction is backward. EDT moves to the left to find **Cairo**, deletes the paragraph preceding the one containing the string, and then resets the direction to forward.

When a search string is followed immediately by ADV or BACK, EDT uses the direction command to determine the direction of that search as well as subsequent editing work. You can rewrite the sample command lines, putting the directions after the search strings.

```
"83" BACK

I.mem^Z  ";" BACK +C D+EL

"Cairo" BACK DPAR ADV
```

If you want to resume the forward direction in the second example after locating the semicolon, you can write the command this way:

```
I.mem^Z ";" BACK ADV C DEL
```

5.7.2 Nokeypad Commands That Affect the Screen Display — REF, SHL, SHR, TOP

The four nokeypad commands that affect the screen image are REF (refresh), SHL (shift left), SHR (shift right), and TOP.

5.7.2.1 REF — When you use REF (refresh), EDT redraws every character from your text that was displayed on the screen, thus eliminating characters that are not part of the text you are editing. These extraneous characters can be notices from the system manager or indications that you have received electronic mail. When you issue the REF command, EDT removes the interrupting text and restores any text characters that were temporarily covered up by the message.

```
for instance, we need to have a thorough review of
New mail from SMIITH s so that we can economize on
items with low usage and high price tags.  Any ideas
  .
  .
  .
```

```
REF
```

```
for instance, we need to have a thorough review of
our purchasing policies so that we can economize on
items with low usage and high price tags.  Any ideas
  .
  .
  .
```

5.7.2.2 TOP – The TOP command puts the current cursor line at the top of your screen. TOP always causes EDT to scroll toward the bottom of the buffer regardless of EDT's current direction. You must have at least 22 lines between the current cursor line and the end of the buffer in order for the TOP command to work. If you issue the TOP command and notice no change on your screen, you can assume that there are less than 22 lines left in the buffer.

You can combine the TOP command with the "move" command to have EDT put a certain entity at the top of the screen. For example, **PAGETOP.** is the default definition for the keypad PAGE function. The keypad function PAGE causes EDT to move to the next page boundary mark and puts that line at the top of the screen.

This example uses the string entity and the TOP command to move the cursor to the **Dear Sir** line and place that line at the top of the screen.

```
"Dear Sir" TOP
```

```
Dear Sir:

        We have an offer for you that you can't refuse.
      For just $10 a day, you can be the proud owner of
        .
        .
        .
```

5.7.2.3 SHL and SHR – SHL (shift left) and SHR (shift right) adjust the screen display horizontally. SHL allows you to shift the visible portion of the current buffer to the left, so that you can view characters located to the right of the current screen width. SHR moves the screen image back to the right. The syntax for SHL and SHR is:

```
[count]SHL

[count]SHR
```

The count specifier refers to the number of horizontal tab stops for EDT to shift the screen image. EDT's preset horizontal tab stops are positioned at eight column intervals. Since SHL and SHR rely only on EDT's preset tab stops (not on the SET TAB value), these commands always move text eight columns. SHR can shift the buffer image only back as far as the default setting. Text cannot be moved farther to the right. If you want to return the screen image to the default state and do not remember how many tab stops you have shifted text to the left, simply give a large count specifier with SHR. EDT will stop the shift when the first column of the buffer is at the left of the screen. For example:

```
1.......2.......3.......4.......5.......6.......7........

2SHL

3.......4.......5.......6.......7.......8.......9........

4SHL

1.......2.......3.......4.......5.......6.......7........
```

5.7.3 Clearing the Search String — CLSS

You can use the CLSS command to clear the search string that is currently in the search buffer. CLSS is most useful in keypad key definitions where the SR (select range) entity is part of the function definition. The two preset keypad definitions that include SR are **CUTSR.** for the CUT function and **CHGCSR.** for the CHNGCASE function.

The SR entity not only refers to a select range, but also references the current search string if there is no active select range and the cursor is located on the current search string. If you press CUT when the cursor is located at the current search string, EDT deletes the search string. With CHNGCASE, EDT changes the case of all letters in the search string when the cursor is located there. You can bypass the search string feature of the SR entity by using the CLSS command in your keypad definitions for these functions.

The following DEFINE KEY commands redefine the CUT function and the CHNGCASE function to clear the search buffer before processing the command:

```
DEFINE KEY 6 AS "CLSS CUTSR."

DEFINE KEY GOLD 1 AS "CLSS CHGCSR."
```

See Chapter 7 for information on how to create key definitions with the line mode DEFINE KEY command.

5.7.4 Sounding the Terminal Bell — BELL

When you issue the BELL command, EDT sounds the terminal bell. This command takes no specifiers and has no effect on the text being edited. (It does not put a CTRL/G in your text.) The BELL command is not affected by SET QUIET.

You can use BELL to have EDT signal when an operation has been completed. Suppose you are looking for a string in a large buffer and want EDT to notify you when it has found the string. First issue the line mode SET QUIET command to turn off the sound if EDT prints the **String was not found** message. Then add the BELL command at the end of the command line, so that EDT will signal if the search is successful.

```
EXT SET QUIET

"Mr. Summers" BELL
```

If EDT is unable to find the string, it does not process the remaining commands on the line. Thus, with SET QUIET in effect, the terminal bell only sounds when the search is successful.

5.8 Formatting Text with the FILL Command

The FILL command in nokeypad mode is similar to the keypad FILL function and the line mode FILL command. All three depend on the current screen width or current SET WRAP value. The SET WRAP value always takes precedence over the screen width. The syntax for the nokeypad FILL command is:

```
[+|-][count]FILL[+|-][count]entity
```

The sign specifiers indicate the direction for the FILL operation. The count specifiers determine how many times EDT will repeat the FILL operation or how many entities will be filled. The entity specifier is generally L, PAR, PAGE, or SR, but you can use other entities.

If SET NOWRAP, the default, is in effect when you issue the FILL command, EDT uses a value of one less than the current screen width setting to determine the maximum line length. Since the valid SET SCREEN widths for screen editing are 80 and 132, EDT will use either 79 or 131.

This example uses the paragraph entity with FILL and uses a SET WRAP value of 50.

```
It is surprising how few companies engage in scientific
market research before deciding to manufacture
and market a new product.
You get the impression that someone high up in the company
started producing an item
and then relied on the marketing staff to create
a demand for it.

EXT SET WRAP 50
FILLPAR

It is surprising how few companies engage in
scientific market research before deciding to
manufacture and market a new product. You get the
impression that someone high up in the company
started producing an item and then relied on the
marketing staff to create a demand for it.
```

5.9 Special Nokeypad Commands — ASC, ^, DATE

Three nokeypad commands enable you to perform special inserts. By using ASC, you can insert any character from the DEC Multinational Character Set in your text. The circumflex (^) enables you to insert the ASCII control characters (DEC Multinational Character Set decimal values 0 - 31). The third command – DATE – causes EDT to insert the date and time at the current cursor location.

5.9.1 ASC (ASCII)

You can use the ASC command to insert any character from the DEC Multinational Character set into your text. EDT puts the character to the left of the current cursor position. The syntax of the ASC command is:

```
[number]ASC
```

The number specifier is the decimal equivalent value for that character. For instance, the decimal value for CTLR/X is 24; for uppercase X it is 88. Appendix G contains a complete list of the DEC Multinational Character Set characters and their corresponding decimal values.

When you use the ASC command without the number specifier, EDT inserts the null character (CTRL/@, decimal value 0), which is displayed as ^@, in your text.

Use the ASC command to put special characters in startup command files or EDT macros when you need to insert control characters in DEFINE KEY or SET ENTITY commands. (See Chapter 7 for information on EDT startup command files, EDT macros, and the DEFINE KEY command. See Chapter 6 for details on the SET ENTITY commands.)

Suppose you want to reverse the keypad functions FNDNXT and FIND on your keypad. You could reset the definitions for these keys in your startup command file. If you are using nokeypad mode to put these commands in a startup command file, you will want to use the ASC command to enter the null characters in the FIND definition. First type the text that you need. Then use the ASC command to insert the null characters in their proper places.

```
I

     DEFINE KEY 11 AS "?'Search for: '."
     DEFINE KEY GOLD 11 AS '"".'
CTRL/Z

-"?"

     DEFINE KEY 11 AS "?'Search for: '."
     DEFINE KEY GOLD 11 AS '"".'

ASC

     DEFINE KEY 11 AS "^@?'Search for: '."
     DEFINE KEY GOLD 11 AS '"".'

"." ASC

     DEFINE KEY 11 AS "^@?'Search for: '^@."
     DEFINE KEY GOLD 11 AS '"".'
```

If you need to insert several of the same special characters next to each other, enclose the ASC command with its number specifier in parentheses and put the repeat count digit(s) before the parentheses. For example, to insert five tab characters (CTRL/I) in a row, use ASC with a repeat count of five.

```
5(9ASC)
```

5.9.2 The Circumflex (^)

The circumflex (^) inserts an ASCII control character (DEC Multinational Character Set decimal values 0 - 31) into your text, as well as into nokeypad command lines. For example, when you use the single-line form of the I (insert) command, you can type circumflex Z (^Z) instead of pressing CTRL/Z. The syntax for the circumflex function is:

```
[count]^[character]
```

The count specifier enables you to insert more than one control character at a time. The character specifier can be any letter from A through Z, the at sign (@), left bracket ([), backslash (\), right bracket (]), the circumflex itself (^), and the underscore (_).

You can use the circumflex to put some CTRL/Gs into your text so that when the file is printed at someone's terminal, the terminal bell will sound. All you have to do is type the circumflex and the letter; then press RETURN:

```
        Happy Birthday to You!
5^G(RETURN)
        Happy Birthday to You!^G^G^G^G^G
```

The bell does not sound during your EDT session. But when the line is printed at the terminal by the operating system, the terminal beeps five times at the end of the line. Neither the circumflexes nor the Gs appear in the printed text. When you use EDT in one of the screen modes, the ^G graphics appear on the screen, so you can delete them if you need to.

5.9.3 DATE

The DATE command inserts the current date and time as supplied by your operating system. Simply type DATE and EDT inserts the information to the left of the cursor regardless of EDT's current direction. The operating system provides the format for the DATE text. Generally the format includes both the date and time with spaces on either side of the DATE text.

```
        DATE:
DATE
        DATE:  10-MAY-1984 16:40:41
```

Once EDT inserts this material, you can treat it as if you had typed the characters yourself. You can use any EDT commands and functions to delete, move, or alter the text.

5.10 Joining Nokeypad Commands on a Single Command Line

Nokeypad mode allows you to use several commands together on a single command line. You have already seen examples of that in this chapter. In addition, you can use parentheses to have a single repeat count affect several commands.

There are three things to be aware of when you put several commands on a single command line:

1. Spaces

   Spaces are not permitted within a single nokeypad command, but you can use them to separate individual commands in a multicommand line. In the examples, the first three command lines produce identical changes to the text; the fourth results in an EDT error message.

```
3SEN D2W IMarch 15, ^Z "wednesday" DW

3SEND2W IMarch 15, ^Z "wednesday"DW

3SEND2WIMarch 15, ^Z"wednesday"DW

3SEN D 2W IMarch 15, ^Z "wednesday" D W
```

The fourth example has spaces separating the D commands from the entities you want to delete. EDT displays the error message **Invalid entity** when it tries to process that part of the command line. EDT thinks you are using the space following the D command as the entity.

2. Errors

   As soon as EDT detects a problem on a multicommand line without parentheses, it stops processing the commands and issues the appropriate error message. The remaining commands on the line are ignored. Thus, in the fourth command line above, EDT only moves three sentences and then prints the error message. It does not see the second mistake, D W, at the end of the command line.

3. Parentheses

   Using parentheses in multicommand lines enables you to group commands together with a repeat count or direction sign. A repeat count preceding a group of commands enclosed in parentheses signals EDT to repeat the entire command sequence that number of times. Similarly, a sign preceding the parentheses tells EDT to temporarily shift to the indicated direction for all commands enclosed in the parentheses.

   On a multicommand line, if a command contained within parentheses fails, EDT does not stop processing the command line. For example, when you are using the "move" command to locate a particular string, you can enclose the command within parentheses. If there are more commands on the line after the "move" command and the string cannot be found, EDT continues to process any commands that follow the right parenthesis on the command line. EDT does not process any commands within the parentheses that follow the "move" command which failed. If you have a repeat count, failure to find the string not only causes EDT to skip over the remaining commands up to the right parenthesis, but also cancels any remaining repeat count within the parentheses.

The following examples show the difference made by parentheses:

```
21.56
35.80
49.95

3(I$^Z BL +V)

$21.56
$35.80
$49.95
```

If you omit the parentheses, the result is:

```
$$$21.56
35.80
49.95

FILE1.RNO;1
FILE2.RNO;3
FILE3.RNO;1
FILE4.RNO;2

4(".." C DEC IMEM^Z DEL)

FILE1.MEM
FILE2.MEM
FILE3.MEM
FILE4.MEM
```

If you omit the parentheses, only the last line is changed:

```
FILE1.RNO;1
FILE2.RNO;3
FILE3.RNO;1
FILE4.MEM
```

Note that the R (replace) command and the multiline form of the I (insert) command can only be placed at the end of a multicommand line. Also, the EPAR and EPAGE entities cannot be used twice in succession as "move" commands. However, by using parentheses, you can create a command such as 3(EPAR +C) to move to the third end of paragraph.

The EXT command can never be placed inside parentheses because it cannot be repeated. If EXT is not the last command in a multicommand line, you must add the line mode CHANGE command as your last line mode command in order for a nokeypad command to follow EXT. For example:

```
EXT SET SCREEN 132 ;CHANGE ;+5PAGE
```

Chapter 6
The SET and SHOW Commands

6.1 Introduction

SET commands allow you to change the way EDT operates. SHOW commands tell you how EDT is operating. All SET commands have a corresponding SHOW command. For example, you can use SET LINES to change the number of lines that can appear on the screen. You use SHOW LINES to find out how many lines the screen is set for.

There are four SHOW commands that have no corresponding SET command. These are SHOW BUFFER, SHOW FILES, SHOW KEY, and SHOW VERSION. The SHOW BUFFER command lists all the buffers that currently exist for your editing session. The SHOW FILES command tells you the name of your input file and output file during your EDT session. The SHOW KEY command gives the current definition for the keypad key name you supply. The SHOW VERSION command displays the current EDT version so that you can enter that information in Software Performance Reports.

All SET and SHOW commands are line mode commands, but many affect the screen modes. Table 6-1 summarizes the functions of each command. In addition to the summary table, this chapter gives details about a few of the commands. For information on SET/SHOW commands, first consult Table 6-1. If you need more details and examples for a particular command, check to see if it is described in detail in this chapter. Otherwise see the appropriate entry in Chapter 8, which describes each SET and SHOW command.

6.2 Summary of SET and SHOW Commands

Table 6-1 contains a brief description of all SET and SHOW commands. The left-most column gives the syntax for the SET commands. Optional words in the command line are listed vertically. You must include one, but only one, optional word in the command line. For example, the SET CASE command can be SET CASE UPPER, SET CASE LOWER, or SET CASE NONE. This convention is also used where appropriate for the SHOW commands, which are listed in the second column.

Whenever a string specifier is required with a SET command, you must surround the string with quotation marks, either single (') or double (").

The third column contains a brief description of the SET command and lists its default, where applicable. When there is no SET command, the SHOW command is described.

The last column tells you which modes are affected by these commands. Although all SET commands are line mode commands, some of them have no effect in line mode.

Table 6-1: SET and SHOW Command Summary

SET Command	SHOW Command	Description	Mode
SET AUTOREPEAT NOAUTOREPEAT	SHOW AUTOREPEAT	Enables EDT to use the DECARM VT100 control sequence that prevents keypad keys from repeating faster than EDT can update the screen. Default: SET AUTOREPEAT (For exceptions, see the appendix for your system.)	K
---	SHOW BUFFER	Lists all buffers that currently exist in your editing session. Indicates the current buffer.	K, L, N
SET CASE UPPER LOWER NONE	SHOW CASE	Flags upper- or lowercase letters on single case terminals. Default: SET CASE NONE	L
SET COMMAND file-spec.	SHOW COMMAND	Allows use of more than one startup command file at the start of an editing session. Default: site-dependent	K, L, N
SET CURSOR top:bottom	SHOW CURSOR	Controls scrolling of the screen relative to the cursor position. Default: SET CURSOR 7:14	K, N
SET ENTITY WORD "string" SENTENCE "string" PARAGRAPH "string" PAGE "string"	SHOW ENTITY WORD SENTENCE PARAGRAPH PAGE	Defines entity boundary delimiters. Defaults: SET ENTITY WORD " <LF><VT><FF><CR>" SET ENTITY SENTENCE ".!?" SET ENTITY PARAGRAPH "<CR><CR>" SET ENTITY PAGE "<FF>"	K, N
---	SHOW FILES	Displays the name of the input file you are editing and the output file that EDT will use if you do not use a file specification with your EXIT command.	K, L, N
SET FNF NOFNF	SHOW FNF	NOFNF suppresses the **Input file does not exist** message when EDT creates a new file. Default: SET FNF	K, L, N

Table 6-1: SET and SHOW Command Summary (Cont.)

SET Command	SHOW Command	Description	Mode
SET HELP file-spec.	SHOW HELP	Determines which version of the HELP file is the current one. Default: SET HELP or SET HELP EDTHELP	K, L, N
---	SHOW KEY [GOLD] keypad-key-number GOLD character [GOLD] CONTROL character [GOLD] DELETE [GOLD] FUNCTION key-number	Prints the definition of the specified keypad mode editing key or key sequence.	K
SET KEYPAD NOKEYPAD	SHOW KEYPAD	Sets the screen mode to either keypad or nokeypad. Default: SET KEYPAD	K, N
SET LINES number	SHOW LINES	Sets the number of lines per screen. The maximum is 22. Default: SET LINES 22	K, N
SET MODE LINE CHANGE	SHOW MODE	Sets the starting mode for EDT. Default: SET MODE LINE	K, L, N
SET NUMBERS NONUMBERS	SHOW NUMBERS	Determines if EDT displays line numbers in line mode. Default: SET NUMBERS	L
SET PARAGRAPH WPS NOWPS	SHOW PARAGRAPH	Determines where EDT puts the cursor when moving to a new paragraph. Default: SET PARAGRAPH NOWPS	K, N
SET PROMPT prompt-type "string"	SHOW PROMPT prompt-type	Determines various prompt characters you can use. Defaults: SET PROMPT LINE "<CR><LF>*" SET PROMPT KEYPAD "<CR>" SET PROMPT NOKEYPAD "<CR>" SET PROMPT HCCHANGE "<CR><LF>C*"	K, L, N

(continued on next page)

Table 6-1: SET and SHOW Command Summary (Cont.)

SET Command	SHOW Command	Description	Mode
SET PROMPT (cont.)		SET PROMPT INSERT "<CR><LF> SET PROMPT INSERTN "<CR><LF>" SET PROMPT QUERY "<CR><LF>"	"
SET QUIET NOQUIET	SHOW QUIET	Determines whether the terminal bell sounds when an EDT message is displayed in a screen mode. Default: SET NOQUIET	K, N
SET REPEAT NOREPEAT	SHOW REPEAT	Determines whether you can use the GOLD keyboard number sequence for repeating keypad functions and for the keypad SPECINS function. Default: SET REPEAT	K
SET SCREEN width	SHOW SCREEN	Sets the number of characters EDT displays on a line. The only width values allowed for VT100-type terminals with advanced video option (AVO) are 80 and 132. The only value allowed for VT100-type terminals without AVO and VT52 terminals is 80. (For VK100 terminals, 84 is the only valid screen width.) Default: Set by operating system.	K, L, N
SET SEARCH GENERAL EXACT WPS CASE INSENSITIVE DIACRITICAL INSENSITIVE SET SEARCH BEGIN END SET SEARCH UNBOUNDED BOUNDED	SHOW SEARCH	Determines how EDT performs searches. Defaults: SET SEARCH GENERAL SET SEARCH BEGIN SET SEARCH UNBOUNDED	K, L, N
SET SUMMARY NOSUMMARY	SHOW SUMMARY	Determines whether summary information is printed when you end your EDT session with the EXIT command. Default: SET SUMMARY	K, L, N

Table 6-1: SET and SHOW Command Summary (Cont.)

SET Command	SHOW Command	Description	Mode
SET TAB number NOTAB	SHOW TAB	Sets the tab size for various tabbing functions. Default: SET NOTAB	K, L, N
SET TERMINAL HCPY VT100 VT52 SET TERMINAL EDIT NOEDIT SET TERMINAL EIGHTBIT NOEIGHTBIT SET TERMINAL SCROLL NOSCROLL	SHOW TERMINAL	Determines how EDT interprets your terminal. Default: characteristics of the terminal the operating system thinks you are using	K, L, N
SET TEXT END "string" PAGE "string"	SHOW TEXT END PAGE	Determines how EDT displays the end of buffer marker or page marker. Defaults: SET TEXT END "[EOB]" SET TEXT PAGE "<FF>"	K, L, N
SET TRUNCATE NOTRUNCATE	SHOW TRUNCATE	Determines whether EDT truncates long lines. Default: SET TRUNCATE	K, N
SET VERIFY NOVERIFY	SHOW VERIFY	Determines whether EDT displays each command in a startup command file or EDT macro as the commands are processed. Default: SET NOVERIFY	K, L, N
---	SHOW VERSION	Tells which EDT version you are using and displays the software copyright information.	K, L, N
SET WORD DELIMITER NODELIMITER	SHOW WORD	Determines how EDT interprets word boundaries. Default: SET WORD DELIMITER	K, N
SET WRAP number NOWRAP	SHOW WRAP	Determines the maximum line length for filled text and for text inserted in keypad mode. Defaults: SET NOWRAP	K, L, N

6.3 SET/SHOW ENTITY

EDT uses a number of entities in the screen modes for a variety of functions and commands. Table 5-1 in Chapter 5, Nokeypad Editing, gives a complete list of all EDT entities. Each entity has certain properties or boundaries that enable EDT to distinguish one entity from another. The SET ENTITY command lets you customize four entities:

WORD	PARAGRAPH
SENTENCE	PAGE

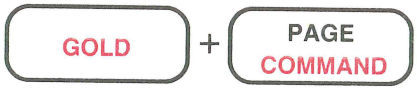
Although SET ENTITY is a line mode command, it is best typed from one of the screen modes if your boundary list includes any control characters. This is because control characters do not appear on your paper or screen when entered from line mode. If your boundary list includes CTRL/M, which references the line terminator, you will need to enter that SET ENTITY command from a screen mode.

If you are including a SET ENTITY command in a startup command file or EDT macro, you can use the keypad SPECINS function or the nokeypad ASC command to enter any control characters that you need. EDT has two ways of displaying control characters. In some instances the control character is shown as the character preceded by a circumflex, for example, ^G for CTRL/G (decimal 7). In other instances, EDT uses angle brackets and letters, for example, <FF> for CTRL/L (decimal 12). (Appendix G shows the symbols EDT uses to display control characters.)

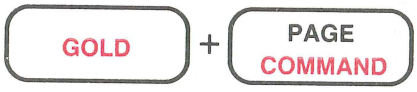
You must include the entity name with the SET ENTITY command phrase, for example, SET ENTITY PAGE. The boundaries, which follow the entity name, are enclosed in quotation marks (single – ' or double – "). You cannot use a SET ENTITY command to add characters to a boundary list. You must include every boundary character that you need each time you issue a new SET ENTITY command.

There are no separating marks or spaces between characters in a boundary list. One follows directly after another. The order of boundary characters is not significant for WORD and SENTENCE, but is for PARAGRAPH and PAGE.

If you type a SET ENTITY command in keypad mode using the COMMAND function, you must press control keys to enter those characters in your boundary list. The control character symbols that EDT displays when you use the COMMAND function are sometimes different from those that EDT displays when you use SPECINS or ASC or when you issue the SHOW ENTITY command. For example, if you include the line terminator in your list, you will see ^M on the **Command:** line, rather than <CR>. However, when you later issue the SHOW ENTITY command, EDT displays <CR>, not ^M.




```
Command:  SET ENTITY WORD  " ^M"
```




```
Command:  SHOW ENTITY WORD
 <CR>
```

In order to include both types of quotation marks in a SET ENTITY boundary list, you must double the type that you have used as the string delimiter for the boundary list. For example, if you are using the single quotation mark as the delimiter for the SET ENTITY WORD command, you must use two single quotation marks within the definition to have EDT interpret the single quotation mark as a word boundary:

```
SET ENTITY WORD              <LF><CR>.,;'':"!'
```

However, when you issue the SHOW ENTITY WORD command to see the word boundary list, EDT displays the quotation mark boundary as a single character.

```
SHOW ENTITY WORD
        <LF><CR>.,;':"!
```

6.3.1 SET ENTITY WORD

A word in EDT is any group of contiguous characters that lies between two word boundary characters. The default word boundary characters are:

space ( )
horizontal tab (or ^I)
linefeed (<LF> or ^J)
vertical tab (<TAB> or ^K)
form feed (<FF> or ^L)
line terminator (<CR> or ^M)

EDT considers each boundary character, except the space, as a word. The space is treated as part of the word it bounds. When you move the cursor word by word, EDT stops at every line terminator, form feed, and so forth. (At the end of this section there is a discussion of the SET WORD NODELIMITER command, which causes EDT to skip over word boundaries and not consider them as words.)

When you issue the SHOW ENTITY WORD command, EDT displays the current word boundary list. The default is:

```
*SHOW ENTITY WORD
        <LF><VT><FF><CR>
```

The blank area to the left of <LF> consists of one space and a horizontal tab character.

When defining your own word boundaries, you might want to make the word entity more literal by separating punctuation marks from the letters themselves. For some applications it might be important to separate only parentheses, brackets, and braces in addition to the default word boundaries. You might want to include the digits 0 through 9 in your list of word boundaries.

Here are some sample SET ENTITY WORD commands as they appear if you use SPECINS or ASC to enter the control characters:

```
SET ENTITY WORD '            <CR>.;?()"'
```

Includes space, horizontal tab, line terminator, and various punctuation characters. This list could be used for a file that you know has no <LF>, <VT>, or <FF> characters.

```
SET ENTITY WORD "            .|$"
```

Includes space, horizontal tab, period, vertical bar, and dollar sign. You might find this useful when working with an inventory or price list.

```
SET ENTITY WORD "        0123456789()<>=+-/_'""*,;$"
```

Includes numbers and punctuation characters that occur in BASIC programs. In order to include both kinds of quotation marks in the boundary list, you must double the kind that you use to enclose the entire list. In this example, the double quotation mark has been doubled. As with the other examples, the space and horizontal tab are included at the beginning of the boundary list.

6.3.2 SET ENTITY SENTENCE

EDT considers the sentence entity to be bounded by a character followed by at least one space or a line terminator. The default characters are the period (.), exclamation point (!), and question mark (?). Unless these appear with spaces after them or at the ends of lines, EDT does not consider the punctuation mark to signal the end of a sentence. When you type a SET ENTITY SENTENCE command, do not include the space or the line terminator indicator in your boundary list.

Here are some sample SET ENTITY SENTENCE commands.

```
SET ENTITY SENTENCE ".?!)}>]"
```

Includes the three default boundaries and adds the closing parenthesis, brace, and brackets. This definition is useful if you have sentences enclosed in parentheses, braces, or brackets because normally EDT does not recognize the period inside one of these marks as the end of a sentence.

```
SET ENTITY SENTENCE '.?!"):-'
```

Includes the three default boundaries and adds quotation marks, closing parenthesis, colon, and hyphen. If you use the hyphen followed by a space to indicate a dash, EDT considers the sentence to end when it encounters such a hyphen. Confusion could be caused if you use the hyphen at the end of a line to break a word.

6.3.3 SET ENTITY PARAGRAPH

The boundary for the paragraph entity is not a list of distinct single-character signals. Rather it is a single boundary that can be composed of more than one character. The default is <CR><CR> which indicates two line terminators in succession. If there is a space on the seemingly blank line that separates two paragraphs, EDT ignores that combination (<CR>␣<CR>) and does not consider it to be a paragraph separator.

If you need to, you can mark paragraph ends with a character combination such as xxx. If you use the RUNOFF text formatter, you will have few blank lines (if any) in your text, but many text separators indicated by **.b**. The **.b** text separator can be used as the paragraph boundary.

When you use letters as a paragraph marker, EDT uses the current search parameters (except BEGIN/END) in locating the next boundary mark. When SET SEARCH EXACT is in effect, the paragraph boundary mark in your text must match the SET ENTITY PARAGRAPH string exactly for both case and diacritical marking. For example, if the SET ENTITY PARAGRAPH string is **.B**, EDT will not recognize **.b** as a paragraph separator because the **b** in your text is lowercase. When EDT locates a boundary mark, it positions itself after the mark so that you can move immediately to the next one, if necessary.

Here are some SET ENTITY PARAGRAPH examples.

```
SET ENTITY PARAGRAPH "zxc<CR>"
```
Sets the end of paragraph mark to be the three letters "zxc" followed by a line terminator.

```
SET ENTITY PARAGRAPH ".b"
```
Sets the end of paragraph mark to be **.b**, the RUNOFF blank line signal. If you do not include the line terminator after this mark, EDT will stop at places where you have embedded the command in a list of other RUNOFF instructions, for example, **.b.nf.nj**, **.els.b.f.nj**, or **.b3**.

```
SET ENTITY PARAGRAPH "<CR>."
```
This is another RUNOFF oriented paragraph marker. It assumes that a paragraph ends as soon as you give a new RUNOFF command. Since all RUNOFF commands must be typed at the beginnings of lines and must begin with a period, this SET ENTITY command stops at every new RUNOFF command line that occurs in your text.

6.3.4 SET ENTITY PAGE

The page entity boundary is like the paragraph marker in that you can establish only a single boundary at a time, but you can use a group of characters as the marker. The default page entity marker is the form feed (<FF>). To insert a form feed character in your text, use CTRL/L.

You can set the PAGE entity definition to be any string of characters. For example, if you use the RUNOFF text formatter, you might want to use the header level marker (.HL) as a page indicator. In other types of text you might find it convenient to use the word **page** itself as the page boundary marker. If you are writing a line numbered program, you might want to have **000**(SP) as a page marker, so that EDT will stop at every line number divisible by a thousand.

As with the paragraph entity, if you use letters in your page boundary mark, EDT relies on the current search parameters in locating the page separator.

Here are some SET ENTITY PAGE commands.

```
SET ENTITY PAGE "PAGE"
```
EDT looks for the word **page** to signal a page boundary.

```
SET ENTITY PAGE ".HL1"
```
RUNOFF headers of level 1 are used as page boundaries.

```
SET ENTITY PAGE "Section "
```
The word **Section**, followed by a space, is the page marker.

6.4 SET/SHOW SEARCH

The EDT search facility uses different parameters to offer a variety of ways to search for strings. These search parameters affect substitution commands as well as "find" operations. There are nine parameters, divided into three groups:

1. Five parameters are concerned with two characteristics of letters in a string: case and diacritical marking.

2. Two parameters deal with the position of the cursor after the search string has been found in the change modes.
3. Two parameters determine the extent of the search in the current or specified buffer.

When you use the SET SEARCH command, you can specify only one of the nine parameters per command, for example SET SEARCH EXACT. The SHOW SEARCH command displays the parameters that are currently in effect for all three groups, for example:

```
*SHOW SEARCH
general begin unbounded
```

6.4.1 Parameters That Match Case and Diacritical Marks

EDT has five ways of handling differences in case and diacritical marks when performing searches.

GENERAL
EXACT
WPS
CASE INSENSITIVE (CI)
DIACRITICAL INSENSITIVE (DI)

SET SEARCH GENERAL is the default. When it is in effect, EDT pays no attention to differences in upper- and lowercase letters in performing a search. EDT also ignores the presence or absence of diacritical marks. EDT considers the strings in each of the following lines to be identical:

```
BASIC Basic basic
MACK MacK Mack mack
DEL Del deL del
AUSTERE Austére austere austére
```

When SET SEARCH EXACT is in effect, EDT matches the search string exactly. Upper- and lowercase letters are considered to be different. Letters with diacritical marks, such as the umlaut (¨), acute accent (´), or tilde (~), will not match letters without such marks or with different marks. If your search string is "The C", EDT finds these words in your text:

```
The CLEAR command
The Cardinals
```

It skips over such phrases as **The cheese**, **the coldest winter**, or **the CHANGE command**.

Under SET SEARCH EXACT, each of the following 14 letters is a distinct character:

```
A   a        À   à        Á   á        Â   â
Ã   ã        Ä   ä        Å   å
```

With SET SEARCH WPS, EDT matches only uppercase letters that are typed in the search string with uppercase letters found in the text. Any lowercase letters given in the search string are matched with either lower- or uppercase letters in the text.

When the search string contains only lowercase letters, WPS works exactly like GENERAL; when the search string has only uppercase letters. WPS is the same as EXACT for case.

Under SET SEARCH WPS, the search string **Read the** matches the first eight characters in all of the following phrases:

```
READ theory                          Read the book
Read THEMES AND VARIATIONS           Read The Telltale Heart
```

It does not match **read The Crucible** or **read the sign**.

SET SEARCH CASE INSENSITIVE (SET SEARCH CI) matches letters regardless of case, but does not ignore differences in diacritical marks. The following three strings match:

```
LEGION          Legion          legion
```

The string **légion** does not match.

SET SEARCH DIACRITICAL INSENSITIVE (SET SEARCH DI) matches the case of letters exactly, but ignores diacritical marks. The following strings match:

```
stuck           stück
```

However, neither **Stuck** nor **Stück** is a match because of the uppercase S.

6.4.2 Cursor Location

The second group of SET SEARCH parameters determines where EDT positions the cursor after the string has been found. This group affects only the change modes. In line mode, EDT is always positioned to the right of the last character in the search string, although you are not aware of the location unless you are processing a line mode search from a change mode.

The two choices in change modes for the cursor position are:

```
BEGIN           END
```

With BEGIN, the default, the cursor moves to the first character of the search string. With END, the cursor moves to the character just after the last character in the string. These positions are not affected by EDT's current direction. The current direction only influences whether EDT searches for the string toward the bottom or top of the buffer.

6.4.3 Search Limits

The last group of SET SEARCH parameters determines the extent to which EDT will search for a string. The two choices are:

```
UNBOUNDED           BOUNDED
```

If the search is UNBOUNDED, which is the default, EDT begins the search at the current or specified location and continues to look until it either finds the string or reaches an end of the buffer. The direction of the search determines whether an unsuccessful search continues through the top or bottom of the buffer. (In most cases, after an unsuccessful search, EDT returns to the position where it was when you issued the command. A notable exception is the SUBSTITUTE NEXT command.)

When BOUNDED is in effect, EDT continues to search forward or backward until it reaches a PAGE boundary. If it has not found the string by that time, EDT displays the **String was not found** message.

If you have no PAGE boundaries in your buffer, there is no difference in effect between BOUNDED and UNBOUNDED. The default PAGE boundary is the form feed (<FF> – CTRL/L). If you use the SET ENTITY PAGE command to establish a different PAGE boundary, EDT uses that new boundary as its end limit for bounded searches.

6.5 SHOW BUFFER

The SHOW BUFFER command has no corresponding SET command because EDT uses many different commands to create buffers during an editing session. EDT automatically creates the MAIN and PASTE buffers when the editing session begins. If you type the SHOW BUFFER command, you always see at least these two buffer names in the list.

Every time you create a new buffer during your editing session, its name is added to the SHOW BUFFER list, even if it has no text in it. When you use the line mode CLEAR command to delete a buffer, the buffer name no longer appears in the SHOW BUFFER list. The MAIN and PASTE buffers are exceptions to this rule. The CLEAR command can only delete the contents of MAIN and PASTE; it cannot delete the buffers themselves.

Besides the names of the buffers, SHOW BUFFER tells you the number of lines in each buffer and shows you which buffer is the current one. The current buffer is marked with an equal sign immediately in front of the buffer name (for example, =MAIN).

Sometimes you will notice an asterisk (*) following the line count for the MAIN buffer. The asterisk indicates that EDT has not had a chance to read through the entire text and has only seen as many lines as are specified. When you first call up a file for editing with EDT and are still in line mode, EDT is only aware of the first line of that file. Thus, if you type SHOW BUFFER as your first command, EDT prints this response:

```
=MAIN    1*        lines
 PASTE   No        lines
```

If you issue the SHOW BUFFER command from keypad mode before you do any editing, EDT prints the following (assuming you have more than 44 lines in the buffer and that your screen is set to 22 lines):

```
=MAIN    44*       lines
 PASTE   No        lines
```

You can enter and edit any buffer that appears on the SHOW BUFFER list. Buffer names that do not appear on the list cannot be entered or edited. These "hidden" buffers include: delete character buffer, delete word buffer, delete line buffer, search buffer, and substitute buffer.

Two keypad functions, REPLACE and SUBS, use a buffer called DELETE. The first time you use one of these functions during your editing session, EDT creates the DELETE buffer. EDT overwrites the contents of DELETE each time you use REPLACE or SUBS. You can enter and edit this buffer at any time (although that will have no effect on the REPLACE or SUBS operation.) Its name appears on the SHOW BUFFER list. Since EDT automatically accesses this buffer whenever you use the keypad REPLACE or SUBS function, it is advisable to leave DELETE for EDT to use. For example, suppose you remove some lines from another buffer and store them in the DELETE buffer for later use. If you use either REPLACE or SUBS in the meantime, the text that you put in the DELETE buffer will be overwritten and thus lost for future use.

Here are some examples of the kind of information you can get from the SHOW BUFFER command.

1. After inserting and deleting some text, you type the SHOW BUFFER command to find out how many lines are in your MAIN buffer.

```
*SHOW BUFFER
=MAIN    28        lines
 PASTE   4         lines
```

2. You have used INCLUDE to bring in an external file containing a list of addresses, but you forgot what buffer name you used with your INCLUDE command. Use the SHOW BUFFER command to find out the name of that buffer.

```
*SHOW BUFFER
 ADDRS   21        lines
=MAIN    86        lines
 PASTE   7         lines
```

3. You have been using EDT to make slightly different copies of the same letter to be mailed to different people. Each letter version is in a different buffer. Now that you are ready to put the letters in external files, type the SHOW BUFFER command to see the names of all the buffers.

```
*SHOW BUFFER
 SANDERS          104      lines
 GRUBER 105       lines
 RESNICK          102      lines
 BRIN    104      lines
 KORNFIELD        103      lines
=MAIN    97        lines
 PASTE   3         lines
```

4. You first use the CLEAR command to delete the buffer named EXTRA. Then you decide to use CLEAR to delete the contents of the PASTE buffer. You can use the SHOW BUFFER command to verify how the CLEAR command works.

```
*SHOW BUFFER
 EXTRA   58        lines
 INCLUDE           168       lines
=MAIN    936       lines
 PASTE   5         lines

*CLEAR EXTRA
*SHOW BUFFER
 INCLUDE           168       lines
=MAIN    936       lines
 PASTE   5

*CLEAR PASTE
*SHOW BUFFER
 INCLUDE           168       lines
=MAIN    936       lines
 PASTE   No        lines
```

6.6 SHOW KEY

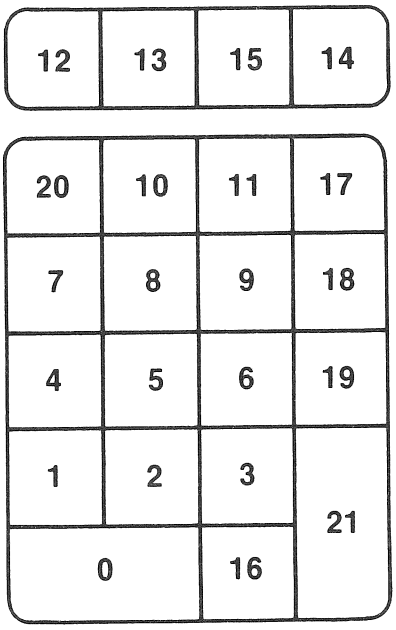
The SHOW KEY command is useful when you are defining keys for keypad editing. Not only can the command be used to refresh your memory about a key you defined, but you can also use it to help you create new key definitions. Tables 7-2 through 7-5 in Chapter 7 show the preset definitions for every keypad mode editing key. You can have those same definitions printed out at the terminal during your EDT session using the SHOW KEY command. (For information on how to define and redefine keypad editing keys, see Chapter 7.)

SHOW KEY is a line mode command, even though the key definitions it shows can only be used in keypad mode. Because of the line mode limitations, you cannot see the key definitions by pressing the keys on the terminal. You must ask for the key definitions in terms that line mode can understand.

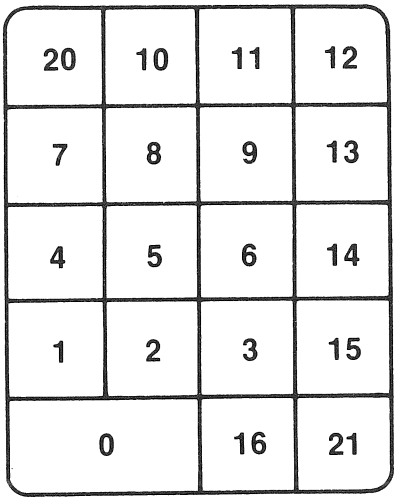
The SHOW KEY command uses the same conventions as the DEFINE KEY command for indicating keypad keys, CTRL key sequences, GOLD key sequences, GOLD/CTRL key sequences, and on LK201 keyboards FUNCTION keys. In brief, type the word GOLD for the GOLD key, the word CONTROL for the CTRL key, the word DELETE for the DELETE key, and the word FUNCTION for LK201 function keys. The definitions for BACKSPACE, TAB, and LINEFEED are found through their control character equivalents (CTRL/H, CTRL/I, and CTRL/J).

You must also use the special keypad numbers that EDT assigns to each keypad key, such as 16 for the keypad period key, or 21 for the ENTER key. Figure 6-1 shows the keypad number equivalents for both the VT100 and VT52 keypads.

Figure 6-1: EDT Keypad Key Numbers



VT100

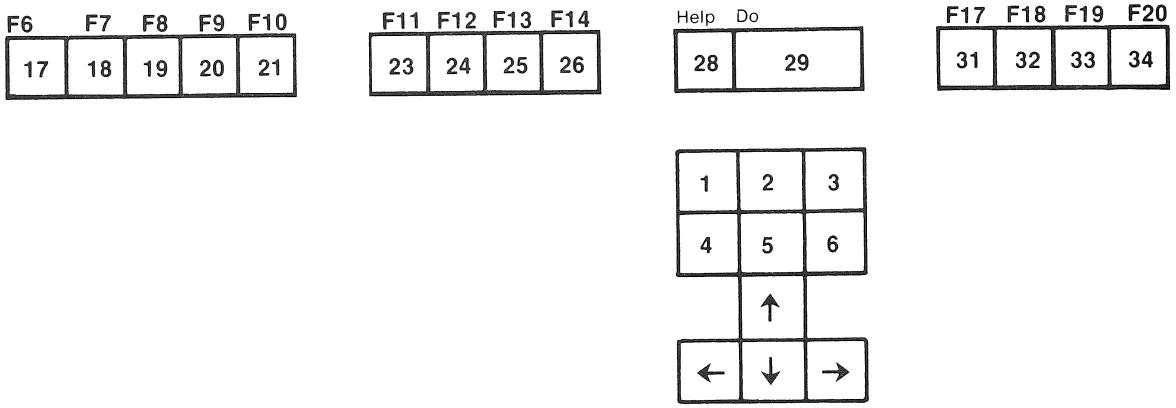


VT52

The FUNCTION keys on the LK201 editing keypad have numbers that you must use in combination with the word FUNCTION when asking EDT to display the definitions of those keys. For the arrow keys, use the same numbers as those for the standard VT100 keypad, for example, SHOW KEY 15 for the Left Arrow key. You can redefine the function keys on the top row of the LK201 keyboard except for the five keys at the left of the row (F1 through F5), which are not shown in the diagram. EDT has preset definitions for two of the remaining function keys – the keys labelled F12 and F13. EDT uses FUNCTION 24 and FUNCTION 25 to identify those keys.

Figure 6-2 shows the FUNCTION key numbers for the LK201's secondary keypad, as well as for the function key row at the top of the keyboard.

Figure 6-2: FUNCTION Key Numbers for the LK201 Keyboard



The following examples show the different ways to express key names in SHOW KEY commands:

```
*SHOW KEY 1
```
Displays the definition for the 1 keypad key.

```
*SHOW KEY GOLD 9
```
Displays the definition of the alternate function for the 9 keypad key.

```
*SHOW KEY 20
```
Displays the definition of the GOLD key.

```
*SHOW KEY 21
```
Displays the definition for the keypad ENTER key.

```
*SHOW KEY GOLD 16
```
Displays the definition of the alternate function for the . keypad key.

```
*SHOW KEY GOLD P
```
Displays the definition for the key sequence GOLD plus P.

```
*SHOW KEY CONTROL G
```
Displays the definition for the key sequence CTRL plus G.

```
*SHOW KEY GOLD CONTROL D
```
Displays the definition for the key sequence GOLD plus CTRL plus D.

```
*SHOW KEY DELETE
```
Displays the definition for the DELETE key.

```
*SHOW KEY CONTROL I
```
Displays the definition for the TAB key (as well as the key sequence CTRL plus I).

```
*SHOW KEY FUNCTION 24
```
Displays the definition for the F12 key on the LK201 function key row.

```
*SHOW KEY FUNCTION 6
```
Displays the definition for the **Select** key on the LK201 editing keypad.

Chapter 7
Advanced EDT Facilities

7.1 Introduction

EDT has some special facilities that enable you to customize and enhance the capabilities of the editor. You can perform most EDT editing functions without using these facilities. But as you become more experienced with EDT, you will find it convenient to use some of these features. The facilities described in this chapter are:

Startup Command Files
Keypad Editing Key Definitions
EDT Macros
Shifting to Different Editing Modes
Editing the Journal File
Tabbing
Hardcopy Change Mode
EDT Line Numbers
HELP File Modification

Each facility is independent of the other. The most frequently used facilities are presented first.

7.2 Startup Command Files

A startup command file contains line mode commands that EDT processes as soon as you call up the editor. You can use startup command files to customize your EDT sessions. Startup command files can be located in your current directory and/or in a central directory.

The default name for most startup command files is EDTINI.EDT (EDT initialization file). System startup command files have the default name EDTSYS.EDT.

Whenever you use EDT to create or edit a file, the editor checks a central directory to see if an EDTSYS.EDT file exists. If one does, EDT processes the contents of that file at the start of the editing session. EDT checks your default directory for a file with the name EDTINI.EDT and processes the commands in that file only if there is no EDTSYS.EDT file in the central location. See the section on using the SET COMMAND and SHOW COMMAND for more information on system startup command files.

If you want EDT to use a another startup command file, that file must have a different file specification. When you are ready to begin editing, include that different file specification in the EDIT/EDT command line that calls up the editor. If you do not include the file type in the specification, EDT assumes that **.EDT** is the file type.

You can have any number of startup command files in your own directory. In order to use a different startup command file, you must include the command file name with your SYSTEM EDIT/EDT command. The system appendixes show how to use a startup command file specification in your EDIT/EDT command line. The general DCL format uses the /COMMAND system qualifier:

```
EDIT /EDT /COMMAND=command-file  file-specification
```

Command-file refers to the specification of your startup command file; **file-specification** refers to the file you want to edit.

If you have several startup command files, give them names that will remind you of their contents. It is not necessary to use the **.EDT** file type, but if you do, you can easily recognize your startup command files. If you intend to have one command file that you use most or all of the time, name it EDTINI.EDT to save retyping the command file specification each time you use EDT.

If you do not specify a command file in your EDT command statement and there is no EDTSYS.EDT file in a central directory and no EDTINI.EDT file in your default directory, EDT starts your session using all the default settings and key definitions.

7.2.1 Creating a Startup Command File

Startup command files can contain any line mode command. The commands most frequently used are:

SET
DEFINE KEY
INSERT
INCLUDE
SHOW
DEFINE MACRO
FIND

To create one of these files, simply use EDT to edit a new file and then type the commands, one command per line. You can use any editing mode to enter the text in a startup command file. If you need to include control characters in some SET commands or key definitions, you should use a screen mode to create the startup command file. Then you can use either the keypad SPECINS function or the nokeypad ASC command to insert the special characters.

Each command must be typed on a single line. You can include comments in your startup command file by preceding the comment or comment line with an exclamation point (!).

If you are planning to put a SET MODE CHANGE command in your file, you might precede it with an exclamation point the first time you use the startup command file. That way, if you have made any typing errors in your command statements, EDT will display these on the screen in line mode so you can correct them. Once you have successfully run the file, simply remove the exclamation point. From then on, your EDT sessions using that startup command file will begin in keypad mode.

7.2.2 Sample Startup Command Files

Here are some examples of startup command files. When you create your own startup command files, you can select commands from among the sample files and add other line mode commands. You can use different files for various types of work. Startup command files can be edited and changed as needed.

The numbers in parentheses to the right of the command text refer you to the brief explanations of these commands. They are not part of the command lines.

Example 1

The first example shows the entire line mode editing session that creates the startup command file. The startup commands themselves appear between the line mode INSERT command and CTRL/Z symbol.

This startup command file contains some SET commands that prepare your session for later editing functions.

```
☞ EDIT /EDT EDTINI.EDT
Input file does not exist
[EOB]
*INSERT
                SET QUIET                       (1)
                SET WRAP 60                     (2)
                SET TAB 5                       (3)
                SET NONUMBERS                   (4)
                SET SEARCH EXACT                (5)
                ^Z
*EXIT
DISK$USER:[BROWN]EDTINI.EDT 5 lines
```

(1) Suppresses the bell sound when EDT prints an error message in screen mode.

(2) Limits the line length for inserting text in keypad mode to 60 characters. Also limits the line length for filling text in all three modes.

(3) Enables you to use all the EDT tab functions at any time. Sets the initial tab size to 5.

(4) Suppresses the EDT numbers from appearing in line mode. EDT still keeps track of the numbers, but does not display them.

(5) EDT matches the case and diacritical marks of letters in search strings exactly as you type the string.

Example 2

This startup command file contains SET commands that you might find helpful if you are working in screen mode at a low data transmission rate. The first line of the file is a brief comment statement describing the nature of the file. The second line simply separates the comment line from the startup commands.

```
! Startup Command File for low data rate sessions.
!
SET LINES 10                    (1)
SET CURSOR 3:7                  (2)
SET MODE CHANGE                 (3)
```

(1) Displays only 10 lines of text on the screen at a time.

(2) Resets the two screen positions at which scrolling takes place.

(3) Starts your EDT session in keypad mode.

Example 3

With this startup command file, EDT starts your editing session in keypad mode and establishes some new key definitions.

```
SET MODE CHANGE                                    (1)
DEFINE KEY GOLD P AS "PAR."                        (2)
DEFINE KEY CONTROL L AS "5NL."                     (3)
DEFINE KEY GOLD W AS "CHGUW."                      (4)
```

(1) Starts your EDT session in keypad mode.

(2) Defines GOLD/P to move the cursor one paragraph.

(3) Defines CTRL/L to move the cursor forward five lines.

(4) Defines GOLD/W to change all lowercase letters in a word to uppercase.

Example 4

This example creates two macros at the start of your session. The macros – EXACT and GENERAL – enable you to change search parameters from **general** to **exact** simply by typing the parameter name.

```
! Creates EXACT and GENERAL macros.
!
FIND =EXACT                                        (1)
INSERT ;SET SEARCH EXACT                           (2)
DEFINE MACRO EXACT                                 (3)
FIND =GENERAL                                      (4)
INSERT ;SET SEARCH GENERAL                         (5)
DEFINE MACRO GENERAL                               (6)
FIND =MAIN                                         (7)
```

(1) Creates a buffer called EXACT and moves there.

(2) Inserts the text **SET SEARCH EXACT**.

(3) Adds **EXACT** to the list of valid line mode commands.

(4) Creates a buffer called GENERAL and moves there.

(5) Inserts the text **SET SEARCH GENERAL**.

(6) Adds **GENERAL** to the list of valid line mode commands.

(7) Returns to the first line of the MAIN buffer.

Example 5

In this example, the startup command file also creates a macro, but does so using the INCLUDE command to enter the macro text from the file TABLE.MAC. The startup command file also starts your session off in keypad mode.

```
! Puts the TABLE macro in your session.
!
INCLUDE TABLE.MAC =TABLE                           (1)
DEFINE MACRO TABLE                                 (2)
```

```
TABLE                                   (3)
FIND =MAIN                              (4)
SET MODE CHANGE                         (5)
```

(1) Puts a copy of the file TABLE.MAC in the buffer called TABLE.

(2) Adds **TABLE** to the list of valid line mode commands.

(3) Processes all the DEFINE KEY commands contained in the TABLE macro.

(4) Returns to the first line of the MAIN buffer so you can start your editing session.

(5) Starts your session in keypad mode.

The macro TABLE, which consists of key definitions, is used as an example in the sections on key definitions as well as the section on creating EDT macros.

Example 6

EDT contains a number of SET and NOKEYPAD commands designed to simulate the WPS (DIGITAL word processing) environment. The following extensive startup command file redefines most of the keypad and keyboard keys on a VT100-type terminal to make it emulate a WPS (word processing) keyboard. Once the startup command file has been processed, EXIT and INCLUDE are the only line mode commands that you have access to. The BELL nokeypad command indicates that a key does not perform the same function as it does on the WPS keyboard.

```
SET NOREPEAT                                          (1)
SET SEARCH WPS                                        (2)
SET WORD NODELIMITER                                  (3)
SET PARAGRAPH WPS                                     (4)
SET NOFNF                                             (5)
SET NOSUMMARY                                         (6)
SET TEXT END "[end]"                                  (7)
SET TEXT PAGE "-New Page-"                            (8)
SET WRAP 72                                           (9)
SET TAB 4                                            (10)
SET MODE CHANGE                                      (11)
DEFINE KEY DELETE AS "DMOV ADV D-C."                 (12)
DEFINE KEY 0 AS "ADV DMOV C."                        (13)
DEFINE KEY 1 AS "BACK DMOV C."                       (14)
DEFINE KEY 2 AS "L."                                 (15)
DEFINE KEY 3 AS "ADV CHGUSR DUPC."                   (16)
DEFINE KEY 4 AS "W."                                 (17)
DEFINE KEY 5 AS "PAR."                               (18)
DEFINE KEY 6 AS "ADV DMOV BELL."                     (19)
DEFINE KEY 7 AS "SEN."                               (20)
DEFINE KEY 8 AS "DMOV BELL."                         (21)
DEFINE KEY 9 AS "ADV DMOV BELL."                     (19)
DEFINE KEY 10 AS "PAGE TOP."                         (22)
DEFINE KEY 11 AS "DMOV ADV DEW."                     (23)
DEFINE KEY 16 AS "DMOV ADV TGSEL."                   (24)
DEFINE KEY 17 AS "DMOV ADV DC."                      (25)
DEFINE KEY 18 AS "DMOV ADV CUTSR."                   (26)
DEFINE KEY 19 AS "DMOV ADV PASTE."                   (27)
DEFINE KEY 21 AS "DMOV '>'+1C."                      (28)
DEFINE KEY GOLD 0 AS "DMOV."                         (29)
DEFINE KEY GOLD 1 AS "."                             (30)
```

```
DEFINE KEY GOLD 2 AS "L."                                                    (15)
DEFINE KEY GOLD 3 AS "ADV CHGLSR DLWC."                                      (31)
DEFINE KEY GOLD 4 AS "W."                                                    (17)
DEFINE KEY GOLD 5 AS "DMOV FILL+PAR."                                        (32)
DEFINE KEY GOLD 6 AS "ADV DMOV BELL."                                        (19)
DEFINE KEY GOLD 7 AS "SEN."                                                  (20)
DEFINE KEY GOLD 8 AS "DMOV BELL."                                            (21)
DEFINE KEY GOLD 9 AS "ADV DMOV BELL."                                        (19)
DEFINE KEY GOLD 10 AS "PAGE TOP."                                            (22)
DEFINE KEY GOLD 11 AS "DMOV ADV UNDW."                                       (33)
DEFINE KEY GOLD 14 AS "."                                                    (30)
DEFINE KEY GOLD 15 AS "."                                                    (30)
DEFINE KEY GOLD 16 AS "DMOV ADV DESEL."                                      (34)
DEFINE KEY GOLD 17 AS "DMOV ADV UNDC."                                       (35)
DEFINE KEY GOLD 18 AS "DMOV ADV CUTSR PASTE."                                (36)
DEFINE KEY GOLD 19 AS "DMOV ADV PASTE."                                      (27)
DEFINE KEY GOLD 21 AS "DMOV '>'+1C."                                         (28)
DEFINE KEY GOLD "'" AS "DMOV ADV CUTSR=DELETE PASTE."                        (37)
DEFINE KEY GOLD '"' AS "DMOV ADV CUTSR=DELETE PASTE."                        (37)
DEFINE KEY GOLD + AS "ADV DMOV BELL."                                        (19)
DEFINE KEY GOLD , AS "ADV DMOV ^@?*'Search for:  '^@."                       (38)
DEFINE KEY GOLD - AS "ADV DMOV BELL."                                        (19)
DEFINE KEY GOLD . AS "DMOV△''."                                              (39)
DEFINE KEY GOLD / AS "DMOV SSEL''."                                          (40)
DEFINE KEY GOLD < AS "ADV DMOV ^@?*'Search for:  '^@."                       (38)
DEFINE KEY GOLD = AS "ADV DMOV BELL."                                        (19)
DEFINE KEY GOLD > AS "DMOV△''."                                              (39)
DEFINE KEY GOLD ? AS "DMOV SSEL''."                                          (40)
DEFINE KEY GOLD A AS "ADV DMOV BELL."                                        (19)
DEFINE KEY GOLD B AS "DMOV ADV ER."                                          (41)
DEFINE KEY GOLD C AS "ADV DMOV BELL."                                        (19)
DEFINE KEY GOLD D AS "ADV DMOV BELL."                                        (19)
DEFINE KEY GOLD E AS "."                                                     (30)
DEFINE KEY GOLD F AS "EXT EXIT."                                             (42)
DEFINE KEY GOLD G AS "DMOV ADV EXT INCLUDE ?*'File Name: '."                 (43)
DEFINE KEY GOLD H AS "HELP."                                                 (44)
DEFINE KEY GOLD L AS "ADV DMOV BELL."                                        (19)
DEFINE KEY GOLD M AS "ADV DMOV BELL."                                        (19)
DEFINE KEY GOLD N AS "ADV DMOV I<FF>^Z."                                     (45)
DEFINE KEY GOLD P AS "ADV DMOV I<FF>^Z."                                     (45)
DEFINE KEY GOLD Q AS "ADV DMOV BELL."                                        (19)
DEFINE KEY GOLD R AS "ADV DMOV BELL."                                        (19)
DEFINE KEY GOLD T AS "BR ADV."                                               (46)
DEFINE KEY GOLD V AS "ADV DMOV BELL."                                        (19)
DEFINE KEY GOLD W AS "DMOV ADV FILLSR."                                      (47)
DEFINE KEY GOLD Z AS "."                                                     (30)
DEFINE KEY GOLD [ AS "ADV DMOV BELL."                                        (19)
DEFINE KEY GOLD \ AS "DMOV ADV DATE."                                        (48)
DEFINE KEY GOLD _ AS "ADV DMOV BELL."                                        (19)
DEFINE KEY GOLD ` AS "ADV DMOV BELL."                                        (19)
DEFINE KEY GOLD { AS "ADV DMOV BELL."                                        (19)
DEFINE KEY GOLD } AS "."                                                     (30)
DEFINE KEY GOLD | AS "DMOV ADV DATE."                                        (48)
DEFINE KEY GOLD ~ AS "ADV DMOV BELL."                                        (19)
DEFINE KEY GOLD DELETE AS "ADV DBL."                                         (49)
DEFINE KEY CONTROL A AS "."                                                  (30)
DEFINE KEY CONTROL D AS "."                                                  (30)
DEFINE KEY CONTROL E AS "."                                                  (30)
DEFINE KEY CONTROL H AS "ADV DMOV BELL."                                     (19)
```

```
DEFINE KEY CONTROL I AS "ADV DMOV TAB."                    (50)
DEFINE KEY CONTROL J AS "ADV DMOV DBW."                    (51)
DEFINE KEY CONTROL K AS "."                                (30)
DEFINE KEY CONTROL L AS "."                                (30)
DEFINE KEY CONTROL R AS "."                                (30)
DEFINE KEY CONTROL T AS "."                                (30)
DEFINE KEY CONTROL U AS "."                                (30)
DEFINE KEY CONTROL W AS "."                                (30)
DEFINE KEY CONTROL Z AS "."                                (30)
DEFINE KEY GOLD CONTROL H AS "ADV DMOV BELL."              (19)
DEFINE KEY GOLD CONTROL J AS "DMOV ADV DBSEN."             (52)
DEFINE KEY GOLD CONTROL M AS "ADV DMOV BELL."              (19)
```

(1) Disables the keypad mode GOLD/repeat and SPECINS functions.

(2) Letters typed in uppercase in a search string must match uppercase letters in the string found in the text.

(3) Default word delimiters are not considered to be words.

(4) When you specify the paragraph entity, EDT puts the cursor on the first character that is not a line terminator.

(5) Suppresses the **Input file does not exist** message.

(6) Suppresses the summary message printed after EDT processes the EXIT command.

(7) Replaces the end of buffer mark ([EOB]) with **[end]**.

(8) Replaces the form feed mark (<FF>) with **–New Page–**.

(9) Sets the maximum line length to 72 characters for inserted and filled text.

(10) Sets the tab size to 4. This affects only the initial indentation of a line.

(11) Starts the editing session in keypad mode.

(12) Defines the **DELETE** key to set the default to move and the direction to forward. Deletes the character to the left of the cursor.

(13) Defines the **0** key on the keypad to set the direction to forward and the default to move. Moves the cursor one character to the right.

(14) Defines the **1** key on the keypad to set the direction to backward and the default to move. Moves the cursor one character to the left.

(15) Defines the **2** key on the keypad to move the cursor to the beginning of the next line, depending on the current direction. (The combination of the **PF1** and **2** keys on the keypad has the same function.)

(16) Defines the **3** key on the keypad to set the direction to forward. Changes the case of letters in the select range to uppercase. Sets the default move to uppercase.

(17) Defines the **4** key on the keypad to move the cursor to the beginning of the next word, depending on the current direction. (The combination of the **PF1** and **4** keys on the keypad has the same function.)

(18) Defines the **5** key on the keypad to move the cursor to the beginning of the next paragraph, depending on the current direction.

(19) Defines the **6** key on the keypad to set the direction to forward and the default to move. Rings the terminal bell. (A number of other key sequences have the same function.)

(20) Defines the **7** key on the keypad to move the cursor to the beginning of the next sentence, depending on the current direction. (The combination of the **PF1** and **7** keys on the keypad has the same function.)

(21) Defines the **8** key on the keypad to set the default to move. Rings the terminal bell. (The combination of the **PF1** and **8** keys on the keypad has the same function.)

(22) Defines the **PF2** key on the keypad to move the cursor to the beginning of the next page, depending on the current direction. Moves the current line to the top of the screen when applicable. (The combination of the **PF1** and **PF2** keys on the keypad has the same function.)

(23) Defines the **PF3** key on the keypad to set the default to move and the direction to forward. Deletes text from the cursor to the end of the word on the right.

(24) Defines the period key (.) on the keypad to set the default to move and the direction to forward. Cancels an active select range or sets one end of a select range if none is active.

(25) Defines the **PF4** key on the keypad to set the default to move and the direction to forward. Deletes the cursor character.

(26) Defines the minus key (–) on the keypad to set the default to move and the direction to forward. Moves the select range text to the PASTE buffer.

(27) Defines the comma key (,) on the keypad to set the default to move and the direction to forward. Inserts the contents of the PASTE buffer to the left of the cursor. (The combination of the **PF1** and comma keys on the keypad has the same function.)

(28) Defines the **ENTER** key on the keypad to set the default to move. Locates the string > and moves the cursor one character to the right. (The combination of the **PF1** and **ENTER** keys on the keypad has the same function.)

(29) Defines the combination of the **PF1** and **0** keys on the keypad to set the default to move.

(30) Defines the combination of the **PF1** and **1** keys on the keypad to have the keypad mode ENTER function. (Several other key sequences have the same function.)

(31) Defines the combination of the **PF1** and **3** keys on the keypad to set the direction to forward. Changes the case of letters in the select range to lowercase. Sets the default move to lowercase.

(32) Defines the combination of the **PF1** and **5** keys on the keypad to set the default to move. Fills the current paragraph.

(33) Defines the combination of the **PF1** and **PF3** keys on the keypad to set the default to move and the direction to forward. Inserts the most recently deleted word to the left of the cursor.

(34) Defines the combination of the **PF1** and period (.) keys on the keypad to set the default to move and the direction to forward. Cancels the select range.

(35) Defines the combination of the **PF1** and **PF4** keys on the keypad to set the default to move and the direction to forward. Inserts the most recently deleted character to the left of the cursor.

(36) Defines the combination of the **PF1** and minus (–) keys on the keypad to set the default to move and the direction to forward. Transfers the select range to the PASTE buffer and puts a copy back in the current location.

(37) Defines the combination of the **PF1** key on the keypad and the single quotation mark key (') on the main keyboard to set the default to move and the direction to forward. Transfers the select range to the buffer called DELETE and inserts the PASTE buffer contents at the current location. (The combination of the **PF1** key on the keypad and the double quotation mark key (") on the main keyboard has the same function.

(38) Defines the combination of the **PF1** key on the keypad and the comma key (,) on the main keyboard to set the current direction to forward and the default to move. Prompts you to enter a search string. Press any keypad key or RETURN to activate the search. (The combination of the **PF1** key on the keypad and the less-than key (<) on the main keyboard has the same function.)

(39) Defines the combination of the **PF1** key on the keypad and the period key (.) on the main keyboard to set the default to move. Using the current direction, locates the current search string. (The combination of the **PF1** key on the keypad and the greater-than key (>) on the main keyboard has the same function.)

(40) Defines the combination of the **PF1** key on the keypad and the slash key (/) on the main keyboard to set the default to move. Using the current direction, locates the current search string and makes it an active select range. (The combination of the **PF1** key on the keypad and the question mark key (?) on the main keyboard has the same function.)

(41) Defines the combination of the **PF1** key on the keypad and the **B** key on the main keyboard to set the default to move and the direction to forward. Moves to the bottom of the buffer.

(42) Defines the combination of the **PF1** key on the keypad and the **F** key on the main keyboard to end your editing session, saving a copy of the MAIN buffer text in an external file.

(43) Defines the combination of the **PF1** key on the keypad and the **G** key on the main keyboard to set the default to move and the direction to forward. Prompts you to enter the name of the file you want inserted in your text. After you press RETURN, inserts a copy of the file above the current line.

(44) Defines the combination of the **PF1** key on the keypad and the **H** key on the main keyboard to access the keypad mode HELP information.

(45) Defines the combination of the **PF1** key on the keypad and the **N** key on the main keyboard to set the direction to forward and the default to move. Inserts a page boundary marker (<FF>). (The combination of the **PF1** key on the keypad and the **P** key on the main keyboard has the same function.)

(46) Defines the combination of the **PF1** key on the keypad and the **T** key on the main keyboard to move the cursor to the top of the buffer. Sets the direction to forward.

(47) Defines the combination of the **PF1** key on the keypad and the **W** key on the main keyboard to set the default to move and the direction to forward. Fills the select range.

(48) Defines the combination of the **PF1** key on the keypad and the backslash key (\) on the main keyboard to set the default to move and the direction to forward. Inserts the current date and time. (The combination of the **PF1** key on the keypad and the vertical bar key (|) on the main keyboard has the same function.)

(49) Defines the combination of the **PF1** key on the keypad and the **DELETE** key on the main keyboard to set the direction to forward. Deletes to the beginning of the line.

(50) Defines the combination of the **CTRL** and **I** keys on the main keyboard to set the direction to forward and the default to move. Moves text to the right of the cursor (including the cursor character) one tab stop to the right.

(51) Defines the combination of the **CTRL** and **J** keys on the main keyboard to set the direction to forward and the default to move. Deletes to the beginning of the word.

(52) Defines the combination of the **PF1** key on the keypad with the **CTRL** and **J** keys on the main keyboard to set the default to move and the direction to forward. Deletes to the beginning of a sentence.

7.2.3 Using SET COMMAND and SHOW COMMAND in a Startup Command File

You can branch from one startup command file to another by using the SET COMMAND command. When you include **SET COMMAND file-specification** in a startup command file, EDT stops at that point in the file and looks for the specified startup command file. If EDT finds the second file, it processes the commands in that startup command file and ignores any remaining commands in the first file. If the second file is not found, EDT continues to process any remaining commands in the first file.

The SET COMMAND command is useful in system- or group-wide startup command files. Suppose that there are some commands that everyone in a group needs. These can be placed at the beginning of the system or group startup command file. The remaining commands in the file are designed for less experienced users. Between the two sets of commands you can include a command such as this:

```
SET COMMAND EDTEXPERT.EDT
```

When EDT reaches the SET COMMAND line in the startup command file, it looks in the current default directory to see if the file EDTEXPERT.EDT exists. If EDT cannot find a file with that name in the current default directory, EDT resumes processing the commands in the system-wide file. At the end of the system-wide file, you can have another SET COMMAND command.

```
SET COMMAND EDTINI.EDT
```

EDT now looks in the current default directory of those users who did not have an EDTEXPERT.EDT file. The commands in an individual EDTINI.EDT startup command file are processed after the system- or group-wide startup command file.

Depending on the needs of your group, you can have only one branch from the system-wide startup command file or several. You can also have a branch from the second file that EDT processes. For instance, if you are an experienced user, you could include a SET COMMAND command at the end of the EDTEXPERT.EDT file to have EDT process some additional commands only needed in one of your subdirectories.

You might want to include a SHOW COMMAND command in each startup command file so that users can verify which startup files are affecting their editing session. SHOW COMMAND displays the name of the startup command file that EDT is currently processing.

This sample file shows one way to place SET COMMAND and SHOW COMMAND commands in a startup command file.

```
DEFINE KEY GOLD L AS "SHL."                    (1)
DEFINE KEY GOLD R AS "SHR."                    (2)
SET COMMAND EDTEXPERT.EDT                      (3)
SET SCREEN 80                                  (4)
SET LINES 12                                   (5)
SET CURSOR 4:8                                 (6)
SET COMMAND EDTINI.EDT                         (7)
SHOW COMMAND                                   (8)
```

(1) Defines GOLD L to perform the SHL (shift left) function.

(2) Defines GOLD R to perform the SHR (shift right) function.

(3) Instructs EDT to search for the startup command file EDTEXPERT.EDT in the current or default directory.

(4) Insures that the screen width is set to 80 characters.

(5) Sets the number of lines displayed on the screen to 12.

(6) Resets the cursor positions at which scrolling takes place to the fourth and eighth lines of the display.

(7) Instructs EDT to search for the startup command file EDTINI.EDT in the current or default directory.

(8) Prints the name of the system-wide startup command file at the start of your session. If you see this name, you know that this startup command file is the only one that EDT has processed.

7.3 Defining and Redefining Keys

This section covers the following topics:

- Keys that can be defined
- Key definitions from nokeypad commands
- Key definition processes in both keypad and line modes
- Samples of the key definition process

Keypad editing uses predefined functions for all keypad keys as well as some keyboard and control keys. Using EDT's key definition facility you can redefine any preset key or assign a definition to a key that has none.

EDT has three ways to incorporate key definitions into your editing session:

1. Including the definition in a startup command file
2. Including the definition in an EDT macro
3. Creating the definition during your EDT session

When you include key definitions in a startup command file, they can become a permanent part of your editing sessions.

Keypad key definitions are based on nokeypad command syntax. The nokeypad command syntax enables you to include several commands in a single definition so that with one key sequence you can have EDT perform a series of operations. Key definitions enable you to use the additional commands and entity specifiers that nokeypad mode has.

7.3.1 Basics

The most important thing to remember about defining and redefining keys is that defined keys can be used only in keypad mode. This point is stressed, because you can define keys in both keypad and line mode. The key definition functions are:

CTRL/K	keypad mode
DEFINE KEY	line mode

The DEFK command in nokeypad mode does not define keys from nokeypad mode. The sole purpose of the DEFK command is to enable you to reassign the key definition function in keypad mode to a key other than CTRL/K.

There are several reasons why you might want to create key definitions. The preset keypad functions use about half of the nokeypad entity specifiers. For example, unless you create new key definitions, you cannot use either the sentence or paragraph entity.

Key definitions that insert text can help you with your work. Suppose you have a word or phrase that you continually type. You can define a key that uses the nokeypad I (insert) command in its single line form to insert that text whenever you need it.

Because you can combine several nokeypad commands into one key definition, many more possibilities for key definitions present themselves. You can create a single key definition to find a word, delete it, and substitute one or more words in its place. Keys can be defined to prompt you to type certain information.

Another important use of key definitions is in accessing nokeypad commands that do not otherwise exist in keypad editing, such as SHL and SHR, CHGL and CHGU, and DATE. You can even define keys to access line mode commands.

7.3.2 What Keys Can Be Defined

All keypad keys can be redefined. In addition, there are many keyboard CTRL, GOLD, and GOLD/CTRL sequences that can be defined. Table 7-1 summarizes the types of keys that can be defined and notes the special characteristics and/or restrictions for each type. Additional operating system restrictions for control keys do not appear in this table. (See the appendix for your operating system to find out which control characters to avoid defining.)

Table 7-1: Definable Keys and Their Characteristics/Restrictions

Key Type	Key Definition Characteristics/Restrictions
Keypad Keys	All keypad keys can be defined.
GOLD Keypad Keys	All GOLD/keypad key sequences can be defined.
Control Keys	CTRL can be used with letter keys as well as [, \ ,], ^, and _. Do not define CTRL/C, CTRL/Q, CTRL/S, or CTRL/X on any system. CTRL/H = BACKSPACE, CTRL/I = TAB, CTRL/J = LINEFEED, and CTRL/M = RETURN
GOLD Control Keys	GOLD/CTRL can be used with letter keys as well as [, \ ,], ^, and _. Do not define GOLD/CTRL/C, GOLD/CTRL/Q, GOLD/CTRL/S, or GOLD/CTRL/X on any system.
GOLD Keyboard Keys	Cannot define GOLD with a digit (keyboard keys 0 through 9) or the keyboard minus sign (–). You must surround !, %, ', and " with quotation marks in the DEFINE KEY command.
DELETE Key	(No restrictions)
GOLD DELETE Key	(No restrictions)
FUNCTION Key	Only available for terminals using the LK201 keyboard. Keyboard keys F1 through F5 cannot be defined.*
GOLD FUNCTION Key	Only available for terminals using the LK201 keyboard. Keyboard keys F1 through F5 cannot be defined.*

*For terminals using the LK201 keyboard, there are user definable keys (UDKs) available in addition to the keys shown in Figure 7-2. When you use these UDKs, it is possible on some terminals to define as many as 100 FUNCTION keys – FUNCTION 0 through FUNCTION 99.

7.3.2.1 Keypad Keys – Most keypad keys have preset functions in EDT. These are described in detail in Chapters 3 and 8. All keypad keys can be redefined including the GOLD key itself. You are advised not to redefine the arrow keys since people are accustomed to these keys moving the cursor in other computer applications. (If you do redefine an arrow key, that key will perform the new function in both keypad *and* nokeypad mode.) The only keypad keys that do not have preset functions are the GOLD arrow combinations on VT100-type terminals. You can create definitions for all four of these key combinations.

7.3.2.2 Control Keys – EDT has preset definitions for the following control keys:

CTRL/A	CTRL/F*	CTRL/K	CTRL/T
CTRL/C	CTRL/H	CTRL/L	CTRL/U
CTRL/D	CTRL/I	CTRL/M	CTRL/W
CTRL/E	CTRL/J	CTRL/R	CTRL/Z

* CTRL/F is preset only on VT52 terminals.

You can redefine all control keys that have been preset by EDT except CTRL/C. You are advised not to redefine CTRL/M (RETURN) or CTRL/Z since most people are accustomed to the preset way these keys work from other computer applications.

Other control keys have functions that are preset by your operating system. The operating system function always takes precedence over the EDT function that you might try to assign to a control key sequence. For example, no matter what definition you assign to CTRL/S, whenever you press the keys, your system puts its "No Scroll" feature into effect. These control key sequences cannot be defined for any operating system:

CTRL/C	interrupt operation
CTRL/Q	resume scrolling
CTRL/S	stop scrolling
CTRL/X	delete input

Check the list in the system appendix for additional control keys that are restricted on your operating system.

7.3.2.3 GOLD Control Keys — When you define GOLD/CTRL/character sequences you are restricted to those characters that are allowed for CTRL/character sequences. For example, GOLD/CTRL/Q has the same system restriction that CTRL/Q has. And, just as you cannot define CTRL/+, you cannot define GOLD/CTRL/+.

When using GOLD and CONTROL, press the GOLD key first, then hold down the CTRL key while you press the keyboard key.

7.3.2.4 GOLD Keyboard Keys — A GOLD/keyboard key refers to the combination of the GOLD key with a key on the main keyboard. GOLD keyboard key sequences can include any keyboard character except the digits 0 through 9 and the minus sign (–).

Four punctuation characters must be surrounded by quotations marks when you use them in the DEFINE KEY command (for example, DEFINE KEY "!" AS "string").

```
!          %          '          "
```

When you are defining one type of quotation mark, use the other type to surround it (for example DEFINE KEY " ' " AS "string").

When GOLD is used with a letter key, EDT ignores the case of the letter. But using the SHIFT key with punctuation marks does make a difference. Thus, you can define both GOLD/` and GOLD/~. When using the shifted character, press the GOLD key, then hold down the SHIFT key and press the keyboard key.

Some GOLD key sequences are preset with the same definition as the corresponding CTRL key sequence. For example, the preset functions for CTRL/A and CTRL/W are the same as those for GOLD/A and GOLD/W. However, you can define CTRL/A and GOLD/A to have different definitions. Redefining one has no effect on the definition of the other. Similarly, CTRL/W and GOLD/W can be defined independently.

7.3.2.5 Special Keyboard Keys — The special keyboard terminal editing keys that have preset EDT functions are:

BACKSPACE	LINEFEED	TAB
DELETE	RETURN	

Note that the LK201 keyboard does not use the labels BACKSPACE, DELETE, or LINEFEED. The BACKSPACE and LINEFEED definitions are assigned to the F12 and F13 keys on the function key row, located across the top of the keyboard. The DELETE key has the symbol ⟨X⟩ on it.

The BACKSPACE, LINEFEED, RETURN, and TAB keys are all linked to control keys.

BACKSPACE	⟷	CTRL/H	LINEFEED	⟷	CTRL/J
TAB	⟷	CTRL/I	RETURN	⟷	CTRL/M

In order to redefine these four special keys using the line mode DEFINE key command, you must redefine their control key equivalents.

```
DEFINE KEY CONTROL J AS "string"
```

When terminals using the LK201 keyboard are operating in VT100 mode, the F12 and F13 keys are linked to CTRL/H and CTRL/J respectively. However, when these terminals operate in VT200 mode, the F12 and F13 keys are *not* linked to any control keys. F12 has the same preset definition as CTRL/H and F13 has the same preset definition as CTRL/J. However, the control functions can be defined independently from the function row keys, just as CTRL/A can be defined independently from GOLD/A.

To redefine the F12 and F13 keys, you must use their FUNCTION key names.

F12	(BACKSPACE)	⟷	FUNCTION 24
F13	(LINEFEED)	⟷	FUNCTION 25

```
DEFINE KEY FUNCTION 24 AS "string"
```

When you redefine the DELETE key with the DEFINE KEY command, type the word DELETE.

```
DEFINE KEY DELETE AS "string"
```

All five special keys can be defined with or without GOLD.

```
DEFINE KEY GOLD CONTROL I AS "string"

DEFINE KEY GOLD DELETE AS "string"
```

You are advised not to define either the DELETE key or the RETURN key since most people are accustomed to the way these keys function from other computer applications.

7.3.3 Parentheses, Periods, and Nokeypad Commands in Keypad Definitions

All but two of EDT's preset editing keys have nokeypad definitions associated with them. (The exceptions are GOLD and RESET.) Since key definitions are useful only in keypad mode, most people who plan to define keys are not familiar with the command syntax of nokeypad mode.

Nokeypad command syntax does not permit spaces within individual commands. If the nokeypad command consists of a command word, a count specifier, an entity, and a buffer specifier, you cannot separate any of these elements from one another with spaces. For example, CUT3PAR=EXTRA is a single nokeypad command that removes three paragraphs from your text and stores them in the buffer named EXTRA. Any spaces within the command would produce unexpected results.

When you create key definitions that use two or more nokeypad commands, you can separate the individual commands from one another with spaces. The spaces are not required, but often they make the definition clearer. If you use the nokeypad EXT (extend) command to create key definitions that access line mode commands, you can include spaces wherever they are permitted by the line mode command syntax. For example, EXT MOVE SELECT TO =SAVE accesses the single line mode MOVE command to relocate the selected lines, putting them in the SAVE buffer. See Chapters 5 and 8 for descriptions of nokeypad commands and specifiers.

You can use the SHOW KEY command to see the definition of any key. The SHOW KEY command displays the preset definition for any key or key sequence until you change the definition. EDT displays the message **No definition** when you ask to see the definition for a key that has no preset definition and has not been defined during your editing session. Once you have redefined a preset key or defined another key, the SHOW KEY command tells you the definition you assigned to that key. More information about the SHOW KEY command appears later in this section as well as in Chapters 6 and 8.

Nokeypad mode has a number of entity specifiers, all of which can be used in key definitions. Table 5-1 lists and describes each nokeypad entity.

With nokeypad command syntax, you can put several commands together in one definition. For example, you can create a single definition that causes EDT to locate a string and then insert some text either before or after the string. You can use count specifiers to repeat all or part of a command line. For example, this definition changes the next four instances of COBOL to COBOL-PLUS:

```
4('COBOL' 5C I-PLUS^Z).
```

The nokeypad commands inside the parentheses tell EDT to move to the string **COBOL**, move five characters to the right, and insert the string **–PLUS** to the left of the cursor. The **4** is used to repeat the entire command four times; the **5** moves the cursor five characters to the right each time the command is repeated.

When you end a key definition with a period, you instruct EDT to process the command(s) as soon as you press the appropriate key sequence. If you omit the period signal, EDT does not process the command until another key has been pressed. The period signal is equivalent to pressing the ENTER key after the command is given. (The preset definition of ENTER is simply the period.)

For example, if you define a key as **10+C**, nothing happens when you press that key. Only after you press ENTER, RETURN, or some other editing key whose definition ends with a period, does the cursor advance 10 places to the right. But if **10+C.** is the definition, the cursor moves 10 places to the right as soon as you press the defined key.

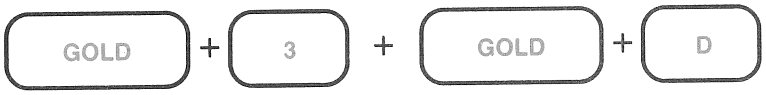
Parentheses, are used to enclose a multicommand line definition so that you can use a repeat count to repeat the entire process. The previous example used parentheses to add the string **–PLUS** to the next four occurrences of the word **COBOL**. If the parentheses had not been there, EDT would have interpreted the number 4 preceding the word COBOL to mean find the fourth occurrence of **COBOL** and add **–PLUS** to just that fourth occurrence.

If you expect to repeat a function a different number of times on different occasions, you can enclose the definition line, except for the period, in parentheses. Whenever you need to issue that function, you can use the keypad mode GOLD/repeat feature to process the function the required number of times.

For example, you can define GOLD/D to be **DPAR** (delete paragraph). Enclose the definition in parentheses and place the period outside.

```
(DPAR).
```

Now, if you want to delete three paragraphs in a row, use GOLD/3 as the repeat count. Remember to enter the count number from the main keyboard, not the keypad.



The three paragraphs, starting with the one in which the cursor is located, have now been deleted.

7.3.4 Using Prompts in Key Definitions

You can define keys to prompt you for input by putting a question mark in your definition followed by the prompt text in quotation marks. Prompts can be used for such things as:

- Nokeypad specifiers – sign, buffer, count, entity, string
- Line mode specifiers – range, buffer, file-specification
- Search strings
- Substitute strings
- File Specifications

When EDT scans the definition line, it first looks to see if there is a period at the end of the command. Next it checks for a question mark prompt. Once EDT collects these two pieces of information, it then checks the other parts of the definition and processes the command.

To use the prompt feature, put a question mark at the place in the command syntax where you want to enter the data from the terminal. It is not necessary to include any prompt text, but if you do, the prompt text reminds you of the type of input EDT is looking for. The prompt text must be enclosed in quotation marks. If you are using the DEFINE KEY command to set up your definition, do not use the same kind of quotation marks for the prompt text that you have surrounding the definition string.

Whenever you type data in response to a prompt, you must press the ENTER key to send that information to EDT so that it will be available when the command is processed. You might find it more convenient to use the RETURN key instead of the ENTER key to transmit the information. If so, place an asterisk (*) immediately after the question mark prompt in your key definition. When EDT sees the asterisk, it allows you to press either RETURN or ENTER to send the data to EDT.

```
DEFINE KEY GOLD S AS "EXT SET SCREEN ?*'Width:  '."
```

If you use the asterisk in the definition, you cannot use the RETURN key to enter CTRL/M as part of your data. This is generally not a disadvantage unless you redefine the FIND function to allow you to press either RETURN or ENTER. In that case, you could not use CTRL/M to include a line terminator in your search string.

Two preset EDT keypad definitions use the question mark prompt feature. These are the COMMAND and FIND functions.

```
COMMAND =    EXT ?'Command: '.

FIND    =    ^@?'Search for: '^@.
```

When you press GOLD/COMMAND, EDT prints the text in the single quotation marks on the command line below your editing work. After you type the line mode command, you press the ENTER key to send the text to EDT. The period at the end of the command line processes the EXT nokeypad command.

The definition for GOLD/FIND is a bit more complex. The @ signs (CTRL/@) surrounding the prompt data are string delimiters, which correspond to the quotation marks used in the nokeypad "move" command. If you are using CTLR/K to create a key definition that requires the null character (^@), you must press the CTRL key and the spacebar to insert ^@ into the definition. Using the null character instead of quotation marks enables you to search for strings that include quotation marks. With this definition, the only character that you cannot locate is the null character.

The question mark in the FIND definition signals EDT to be ready for you to type some data at the terminal. Single quotation marks surround the prompt text. For this function, you can press any keypad key to send the search string to EDT. The period at the end of the definition processes the "move" command as soon as you have completed sending the string text to EDT.

Note that if you are processing a prompt function by pressing a key whose definition begins with either ADV or BACK, EDT processes both the prompt function *and* the direction function of the key you pressed to process the prompt.

Because EDT uses the question mark to signal a prompt, whenever you need to use a question mark in some other way in your definition, you must enter two question marks for every one that you need. For example, if you are using a question mark in a search string, you must type two of them in your key definition.

```
DEFINE KEY GOLD D AS "EXT SUBSTITUTE/year??/1985/ WHOLE."
```

This key definition accesses the line mode SUBSTITUTE command to search for the string **year?** and then replace it with **1985** throughout the buffer. When you display this definition with the SHOW KEY command, EDT prints both question marks. After you press GOLD/D, EDT scans the definition line and determines that the question marks are not a prompt signal.

When you need to use quotation marks in a DEFINE KEY definition that includes a prompt, you will need to double them. The DEFINE KEY command requires that you enclose the definition in quotation marks. Suppose you then use the other type of quotation marks to enclose a prompt string or a search string in the definition. In order to use quotation marks elsewhere in the definition, you must double them. For example,

```
DEFINE KEY GOLD W AS "S/""/?'REPLACE: '/."
```

This nokeypad substitute command searches for a double quotation mark (") and then prompts you to replace it with the character(s) you type.

Here are some definitions using the prompt feature:

```
DEFINE KEY GOLD L AS "CUTL=?'Buffer name: '."
```

Defines GOLD/L to delete the current line and place it in the buffer name that you supply when EDT displays the prompt **Buffer name:**. Press ENTER to send the buffer name to EDT.

```
DEFINE KEY CONTROL D AS "D?'Entity: '."
```

Defines CTRL/D to delete whatever entity you supply when EDT displays the prompt **Entity:**. Press ENTER to send the entity to EDT.

```
DEFINE KEY CONTROL N AS "?'Nokeypad Command: '."
```

Prompts you for a nokeypad command. As soon as you type the command and press ENTER, EDT processes it.

```
DEFINE KEY GOLD 21 AS "S^@?'Find: '^@?'  Substitute:  '^@."
```

Redefines the keypad GOLD/SUBS function so that you can enter both the search and substitute strings as EDT is processing the command. First EDT prompts you for the search string by displaying the prompt **Find:**. As soon as you type the search string and press ENTER, EDT displays the second prompt **Substitute:** next to the string you just typed. Now type the substitute string and press ENTER to send that data to EDT. The ^@s (CTRL/@) are the delimiters for the nokeypad S (substitute) command. If you use ^@ as your delimiter, you can use any character in the DEC Multinational Character Set except the null character (decimal value 0) either in the search or in the substitute string. In order to use ^@ as the delimiter, you must create this definition in a change mode. If you are using the COMMAND or CTRL/K function in keypad mode, you can press the CTRL/spacebar key sequence to insert the ^@ characters. If you are putting the definition in a startup command file or EDT macro, use the keypad SPECINS function or the nokeypad ASC command.

```
DEFINE KEY GOLD I AS "EXT INCLUDE ?'Input File: '."
```

Defines GOLD/I to access the line mode INCLUDE command. When EDT prompts with **Input File:**, type the name of the file you want inserted into your text and then press ENTER.

```
DEFINE KEY GOLD W AS "EXT WRITE ?'Output File: ' SELECT."
```

Defines GOLD/W to put a copy of the select range in an external file. When EDT prompts with **Output File:**, type the name that you want the file to have; then press ENTER.

```
DEFINE KEY GOLD 7 AS "EXT ?*'Command: '."
```

Redefines the preset COMMAND key using the asterisk after the question mark prompt. Now you can press either RETURN or ENTER to process the line mode command.

EDT can only display prompt strings that you include as part of a key definition. EDT has no facility to define keys to print messages at the bottom of the screen.

7.3.5 The Key Definition Process

There are two different ways to define keys: (1) the keypad CTRL/K function and (2) the line mode DEFINE KEY command. CTRL/K can only be used while you are working in keypad mode. The DEFINE KEY command allows you to put key definitions in startup command files and EDT macros, as well as create key definitions during your EDT session.

7.3.5.1 Keypad Mode: CTRL/K – Pressing CTRL/K starts up EDT's key definition process from keypad mode. The steps are:

1. Press CTRL/K.
2. Press the key or key sequence you are defining or redefining.
3. Type the key definition.
4. Press the ENTER key to complete the definition process.

1. As soon as you press CTRL/K, EDT prompts you for the key input as follows:

```
Press the key you wish to define
```

2. Now press the key or key sequence you are defining. Only when you use CTRL/K to redefine the primary function of a keypad key or an LK201 FUNCTION key, do you press a single key. Generally, you will press two or even three keys.

GOLD	+	keypad key
CTRL	+	keyboard key
GOLD	+	keyboard key
GOLD	+	CTRL + keyboard key
GOLD	+	SHIFT + keyboard key
GOLD	+	FUNCTION key (LK201 only)

Whenever CTRL or SHIFT is used, be sure to hold that key down while pressing the keyboard key, just as you would when using the key to perform the defined operation. If you press CTRL or SHIFT and then release that key before pressing the next one, EDT ignores the CTRL or SHIFT and usually prints the message: **That key is not definable**.

3. As soon as you have pressed the key or key sequence you want to define, EDT prints this message:

```
Now enter the definition terminated by ENTER
```

There are three ways to enter your keypad definitions.

- Using nokeypad commands.
- Pressing predefined keypad keys.
- Using a combination of nokeypad commands and predefined keys.

4. When you finish your definition, press the ENTER key to complete the define key process. You are now ready to use the newly defined key or key sequence for the remainder of your current EDT session.

One way to get the feel of creating key definitions is to consolidate several preset keypad functions into one key definition. Tables 7-2 through 7-5 show you the definitions for each EDT preset keypad function. If you find yourself pressing a series of two or three keys repeatedly to perform some function, you can use the tables to put those key definitions together so that one key does the work of several.

Table 7-2: Preset Keypad Key Definitions

EDT Key Name	Keypad Number		Definition	Keypad Key	
	VT100	VT52		VT100	VT52
LINE	0	0	L.	0	0
GOLD OPEN LINE	0	0	(^M-C).	0	0
WORD	1	1	W.	1	1
GOLD CHNGCASE	1	1	CHGCSR.	1	1
EOL	2	2	EL.	2	2
GOLD DEL EOL	2	2	D+EL.	2	2
CHAR	3	---	C.	3	---
GOLD SPECINS	3	14	ASC.	3	→
ADVANCE	4	4	ADV.	4	4
GOLD BOTTOM	4	4	ER.	4	4
BACKUP	5	5	BACK.	5	5
GOLD TOP	5	5	BR.	5	5
CUT	6	3	CUTSR.	6	3
GOLD PASTE	6	3	PASTE.	6	3
PAGE	7	7	PAGETOP.	7	7
GOLD COMMAND	7	7	EXT ?'Command: '.	7	7
SECT	8		(16L).	8	
GOLD SECT		13			↓
GOLD FILL	8	---	FILLSR.	8	---
APPEND	9		APPENDSR.	9	
GOLD APPEND		15			←
GOLD REPLACE	9	12	CUTSR=DELETE PASTE.	9	↑

(continued on next page)

Table 7-2: Preset Keypad Key Definitions (Cont.)

EDT Key Name	Keypad Number		Definition	Keypad Key	
	VT100	VT52		VT100	VT52
HELP	10	10	HELP.	PF2	red
GOLD HELP	10	10	HELP.	PF2	red
FNDNXT	11	8	"".	PF3	8
GOLD FIND	11	8	^@?'Search for: '^@	PF3	8
↑	12	12	-V.	↑	↑
↓	13	13	+V.	↓	↓
→	14	14	+C.	→	→
←	15	15	-C.	←	←
SELECT	16	16	SEL.	.	.
GOLD RESET *	16	16	RESET	.	.
DEL L	17	11	D+NL.	PF4	gray
GOLD UND L	17	11	UNDL.	PF4	gray
DEL W	18	9	DEW.	-	9
GOLD UND W	18	9	UNDW.	-	9
DEL C	19	6	D+C.	,	6
GOLD UND C	19	6	UNDC.	,	6
GOLD #	20	20	GOLD	PF1	blue
GOLD GOLD #	20	20	GOLD	PF1	blue
ENTER	21	21	.	ENTER	ENTER
GOLD SUBS	21	21	(CUTSR=DELETE PASTEKS"").	ENTER	ENTER

* When you are using CTRL/K to define a key, you cannot press a key that has **RESET** as its current definition in order to enter that string in your key definition. Use the key with RESET as its definition to delete the definition line during the CTRL/K operation.

#You cannot use CTRL/K to redefine a key that has **GOLD** as its current definition.

Table 7-3: EDT Preset CTRL (control) and GOLD Key Definitions

CTRL Key Sequence	GOLD Key Sequence	Definition
CTRL/A	GOLD/A	TC.
CTRL/D	GOLD/D	TD.
CTRL/E	GOLD/E	TI.
CTRL/F (VT52 only)		FILLSR.
CTRL/H		BL.
CTRL/I		TAB.
CTRL/J		DBW.
CTRL/K		DEFK.
CTRL/L		^L. (<FF>)
CTRL/M		^M. (<CR>)
CTRL/R	GOLD/R	REF.
CTRL/T	GOLD/T	TADJSR.
CTRL/U**	GOLD/U	DBL.
CTRL/W	GOLD/W	REF.
CTRL/Z	GOLD/Z	EX.

When you are using CTRL/K to define a key, you cannot press the CTRL/U key to enter the string **DBL. into your definition. Either type the string or press GOLD/U. Press CTRL/U to cancel the CTRL/K operation.

Table 7-4: EDT Preset Keyboard Key Definitions

Keyboard Key Name	Definition
BACKSPACE (CTRL/H)	BL.
DELETE***	D–C.
LINEFEED (CTRL/J)	DBW.
RETURN (CTRL/M)	^M. (<CR>)
TAB (CTRL/I)	TAB.

***When you are using CTRL/K to define a key, you cannot press the DELETE key to enter the string **D–C** into your definition. You must type the string or press another key that has D–C. as its definition. Use the DELETE key to edit the definition line during the CTRL/K operation.

Table 7-5: EDT Preset Definitions for Keys and Key Sequences on the LK201 Keyboard (See Table 7-2 for definitions of the numeric keypad keys.)

Key Name	FUNCTION Number	Definition
⟨X⟩ ***	–	D–C.
Do	FUNCTION 29	.
F12 (CTRL/H)	FUNCTION 24	BL.
F13 (CTRL/J)	FUNCTION 25	DBW.
Find	FUNCTION 1	^@?'Search for: '^@
Help	FUNCTION 28	HELP.
Insert Here	FUNCTION 2	PASTE.
Next Screen	FUNCTION 5	(–16L).
Prev Screen	FUNCTION 6	(+16L).
Remove	FUNCTION 3	CUTSR.
Select	FUNCTION 4	SEL.

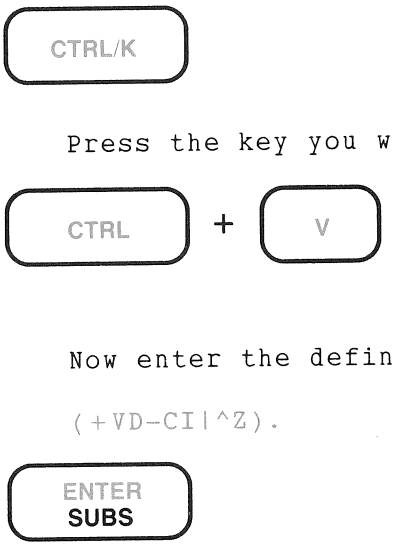
***When you are using CTRL/K to define a key, you cannot press the ⟨X⟩ key to enter the string **D–C** into your definition. You must type the string or press another key that has D–C. as its definition. Use the ⟨X⟩ key to edit the definition line during the CTRL/K operation.

Suppose you are creating a table and need to delete a vertical row of spaces or other characters. You notice that you keep pressing the down arrow and then DEL C. Referring to Table 7-2, you find that **+V** is the definition for the down arrow. **D+C.** is the definition for DEL C. When you combine these two definitions into one, **+VD+C.** is the result. By enclosing the command in parentheses, you can use the keypad GOLD/repeat feature whenever you need it.

```
(+VD+C).
```

Notice that there is only one period used with this command and that it is placed outside the parentheses. If the period is omitted, nothing happens when you press the key until EDT receives another command. Placing the period inside the parentheses causes EDT to print the message **Invalid command** when you press the next key.

Another key definition that comes in handy when you are making tables is one that draws vertical lines. If you try to create a vertical line in keypad mode, you notice that you first press the down arrow to move the cursor down a line. Then you use the DELETE key to remove the space to the left of the cursor, and finally you type the vertical bar (|). Using Table 7-2, you see that **+V** is the definition for the down arrow and that **D–C.** is the definition for the DELETE key. Checking the nokeypad syntax for inserting text, you find that **I** | **^Z** will place a vertical bar character in your text. When you create this definition, you have to type most of the text, but you can press the down arrow key to insert **+V** in the definition.



You must type the DELETE key definition – D-C – instead of pressing the DELETE key. EDT imposes this restriction so that you can use the DELETE key to edit the definition text as you are typing it.

Other editing keys that you cannot press to create keypad definitions are:

BACKSPACE	LINEFEED
RETURN	TAB

If you press one or more of these keys in creating your key definition, EDT inserts the control code for these keys into your key definition. When you try to use the newly defined key, EDT prints the message **Invalid subcommand**. To use these functions in your key definitions, you must enter their default definitions in your definition text: **BL** for BACKSPACE, **DBW** for LINEFEED, **^M** for RETURN, and **TAB** for TAB.

In creating key definitions, do not use the keypad keys to enter characters in the definition text. For example, when you type the period in your key definition, be sure to use the keyboard period key. If you include a repeat count in your definition, always use the keyboard digits to enter the number, not the keypad ones.

The GOLD key cannot be redefined using CTRL/K. You must use the DEFINE KEY command:

```
DEFINE KEY 21 AS "string[.]"
```

The following restrictions apply to certain keys when you are creating definitions under CTRL/K. If a key has **RESET** as its current definition, you cannot press that key to enter the code RESET into the definition line. If you want to define a key to have the RESET function, simply type the word RESET (without the period).

Similarly, you cannot press the DELETE key or CTRL/U to enter D–C. or DBL. into your CTRL/K definition line. In the case of the DBL. definition, you can either type the characters, press GOLD/U, or press some other key that has DBL. as its definition. For D–C., either type the characters or press a key that has D–C. as its definition. EDT has these restrictions: (1) so that you can use the DELETE key to edit your definition line, (2) so that you can use CTRL/U to cancel the CTRL/K process, and (3) so that you can use a key defined as RESET (the preset definition of RESET) to delete the definition line.

7.3.5.2 Editing and Testing Your CTRL/K Definitions – There are several ways to edit or revise key definitions when you are using CTRL/K:

DELETE key
RESET function
CTRL/U function

Use the DELETE key to make minor edits in the text. Keep pressing DELETE until you reach the character you want to change. Then resume typing the definition.

Pressing GOLD/RESET deletes all the text on the definition line so that you can retype the definition.

You can use CTRL/U to cancel the definition process. CTRL/U causes the cursor to return to its former position in the text buffer. In order to resume your key definition, you must press CTRL/K and start all over again.

If you use CTRL/U to cancel the definition process, the key you were defining will retain its previous definition. However, if you use GOLD/RESET to delete the definition text and then press ENTER without typing a new definition, EDT assumes that you want the key to have no definition.

Once you have defined a key, there is no way to make minor edits in the existing definition. You must redefine the key to revise its definition. For example, if you want to add parentheses to a definition, you must redefine the key from scratch.

If you plan to use a repeat count with a definition, you might want to test the definition first to be sure it performs exactly the way you expect. Putting parentheses around the command, with no repeat count, enables you to test it and then enter the repeat count whenever you use that key with GOLD/repeat.

You can enter nokeypad mode to test the nokeypad commands before using them in a definition. In this way you can experiment with several alternative commands and specifiers. It is possible to define a key to access nokeypad mode from keypad mode so that you can test your definitions without leaving keypad mode. See Example 5 in the section on Key Definitions at Work.

7.3.5.3 Line Mode: DEFINE KEY – The syntax for the line mode DEFINE KEY command is different from CTRL/K. The end result, however, is the same. In line mode, EDT does not display any prompt messages. The entire input for each definition is typed on one line. The general syntax is:

```
DEFINE KEY key-name AS "string[.]"
```

Key-name is the name of the key or its keypad number. The definition, represented by **string**, must be typed completely in nokeypad syntax. Not even arrow keys can be pressed in DEFINE KEY definitions. And the definition must be enclosed in quotation marks (either single – ' or double – ").

Key names must be entered in terms that line mode can understand. You must type CONTROL for the CTRL key and GOLD for the GOLD key when they are used in key sequences. (If you are redefining the GOLD key itself, use the number 20 to identify it.) You must type the word FUNCTION when defining function keys that are available on the LK201 keyboard. However, press the SHIFT key when accessing a shifted character such as the plus sign (+); do not type the word SHIFT.

Here are examples of each kind of DEFINE KEY key name:

```
DEFINE KEY 16 AS "string[.]"
DEFINE KEY GOLD 20 AS "string[.]"
DEFINE KEY CONTROL A AS "string[.]"
DEFINE KEY GOLD CONTROL L AS "string[.]"
DEFINE KEY GOLD ? AS "string[.]"
DEFINE KEY GOLD '!' AS "string[.]"
DEFINE KEY DELETE AS "string[.]"
DEFINE KEY GOLD DELETE AS "string[.]"
DEFINE KEY FUNCTION 23 AS "string[.]"
DEFINE KEY GOLD FUNCTION 2 AS "string[.]"
```

The string specifier refers to the definition. The optional period is the period that completes the processing of the defined command. Most of the time you will want to add the period at the end of your definition. Be sure to put the period inside the ending quotation marks that set off the definition text, but outside the closing parenthesis, if you are using parentheses to enclose your definition.

Notice the quotation marks surrounding the exclamation point (!) in the sixth sample line. When you define a key sequence of GOLD with !, %, ', or ", remember to surround the key name character with quotation marks.

If you are using the DEFINE KEY command interactively during an EDT session, you can use the DELETE key to edit the command line. Simply press DELETE until no more unwanted characters remain on the line. Then type the remainder of the command. When you are creating DEFINE KEY commands for a startup command file or an EDT macro, you can use any EDT commands or functions to edit the command line.

Since you cannot press keypad keys in line mode commands, you must use the special EDT keypad numbers to designate the keypad keys. Figure 7-1 has diagrams of the VT100 and VT52 keypads, showing the keypad key numbers. These special keypad numbers are used in the **key-name** specifier with or without GOLD, depending on the key you want to redefine. For example, to redefine the PAGE key, type:

```
DEFINE KEY 7 AS "-W."
```

The RESET key is referred to as GOLD 16. To redefine it, type:

```
DEFINE KEY GOLD 16 AS "SC."
```

The only time that you will use **20** for the GOLD key is when you are redefining GOLD itself to take another function. If you are redefining the alternate function of the GOLD key, you need to use **GOLD 20** in your DEFINE KEY command. For example:

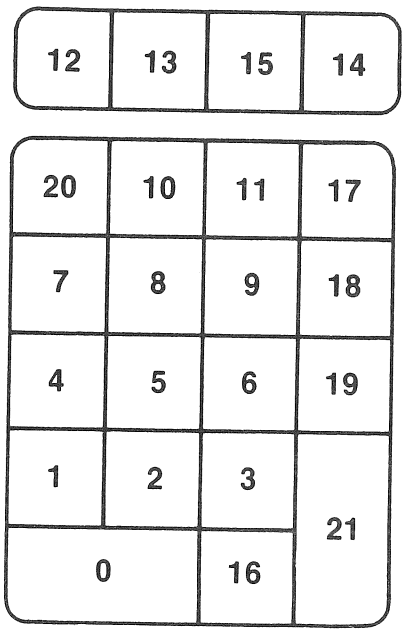
```
DEFINE KEY 20 AS "HELP."

DEFINE KEY GOLD 20 AS "HELP."
```

The SHOW KEY command also uses these special key numbers. For example, to find out the definition for GOLD/SUBS, type:

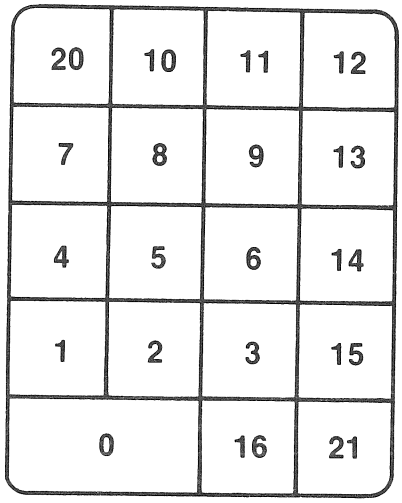
```
SHOW KEY GOLD 21
```

More information on SHOW KEY appears in Chapters 6 and 8.

Figure 7-1: Keypad Key Numbers for Use with DEFINE KEY and SHOW KEY



VT100



VT52

BACKSPACE, LINEFEED, RETURN, and TAB require special handling in the DEFINE KEY command. Line mode does not recognize these key names, although it does recognize DELETE. In redefining them, you must use their CTRL key equivalents on VT100-type and VT52 terminals.

BACKSPACE	→	CONTROL H
LINEFEED	→	CONTROL J
RETURN	→	CONTROL M
TAB	→	CONTROL I

To use these keys with the DEFINE KEY and SHOW KEY commands, type DEFINE KEY CONTROL H AS "string", SHOW KEY CONTROL J, and so forth.

DELETE can be defined with the DEFINE KEY command and have its definition displayed by the SHOW KEY command. It has no control equivalent or special number, so you must type the word DELETE in your command as the key name.

The FUNCTION key numbers for the LK201 keyboard are used with both the upper six keys on the secondary editing keypad and the function key row on the top of the keyboard. Figure 7-2 shows the additional predefined editing keys on the LK201 keyboard.

Figure 7-2: Additional Keypad Editing keys – LK201 Keyboard

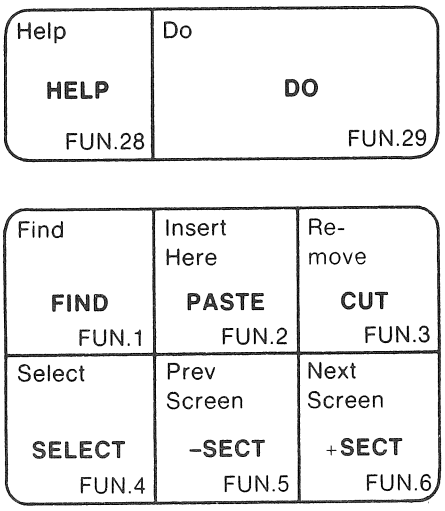
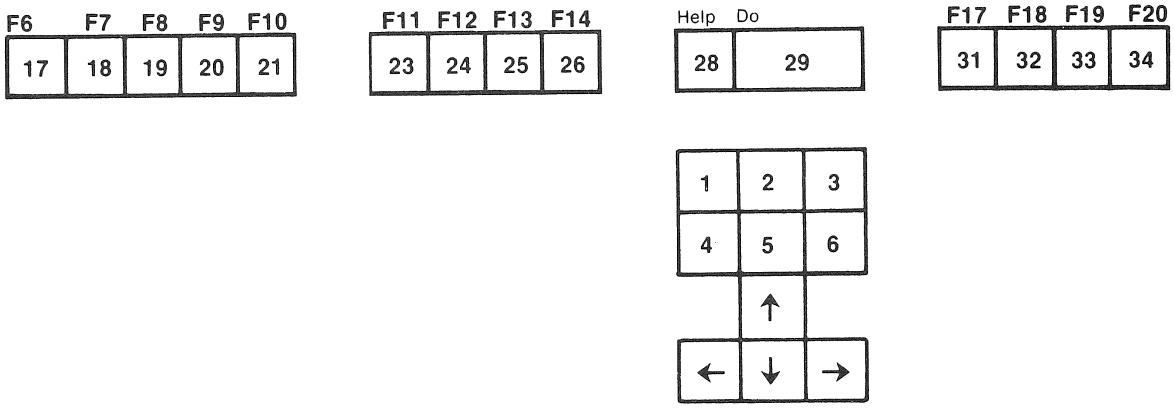


Figure 7-3 shows the FUNCTION key numbers for the LK201 keyboard.

Figure 7-3: FUNCTION Key Numbers for the LK201 Keyboard



For terminals with the LK201 keyboard, there is no key labelled BACKSPACE, DELETE, or LINEFEED. The key with the symbol ⟨X⟩ corresponds to the DELETE key. When you redfine the ⟨X⟩ key use DELETE for the key name: DEFINE KEY DELETE AS "string".

Use **CONTROL M** to redefine the RETURN key and **CONTROL I** to redefine the TAB key.

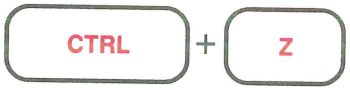
When a terminal with an LK201 keyboard is operating in VT100 mode, the F12 key is linked to CTRL/H and the F13 key to CTRL/J, just as the BACKSPACE and LINEFEED functions are on other VT100-type terminals. When you redefine CTRL/H (BACKSPACE), you are redefining F12. Similarly, if you redefine CTRL/J (LINEFEED), you are redefining F13.

However, if your terminal is operating in VT200 mode, the F12 and F13 keys are independent of their CTRL key counterparts. F12 and CTRL/H still have the same preset definition (BL.), but each can be defined independently of the other, just as CTRL/A can take on a different definition from GOLD/A. F13 and CTRL/J are also independent of each other. To redefine F12 use FUNCTION 24 as the key name; for F13, use FUNCTION 25.

When you create a DEFINE KEY definition that inserts text using the nokeypad I (insert) command, you have to be careful about entering the CTRL/Z signal in the definition string. Depending on which mode you are typing the definition from and under what circumstances, you either type ^Z (circumflex followed by the letter Z), use the keypad SPECINS function or nokeypad ASC command, or press CTRL/Z.

If you are typing the DEFINE KEY command from keypad mode using the GOLD/COMMAND function, you can press CTRL/Z to end the nokeypad I command. EDT will display ^Z on the screen.

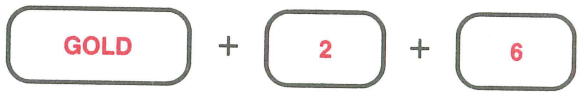
```
DEFINE KEY GOLD R AS "IRhode Island
```

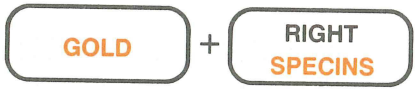



```
DEFINE KEY GOLD R AS "IRhode Island^Z
```

However, if you are typing the DEFINE KEY command as part of the text for a startup command file or EDT macro, you cannot press CTRL/Z.

```
DEFINE KEY GOLD R AS "IRhode Island
```






```
DEFINE KEY GOLD R AS "IRhode Island^Z
```

When you are typing the DEFINE KEY command in line mode and need to use CTRL/Z to complete the nokeypad I command, you must type ^Z (circumflex followed by the letter Z).

Here are some examples of the DEFINE KEY command with completed definitions:

```
DEFINE KEY GOLD _ AS "60(I_^Z)."
```

Defines GOLD/_ to insert a line of 60 underscores in your text. Can be used to draw horizontal lines in your text.

```
DEFINE KEY CONTROL D AS "(+V D+C)."
```

Defines CTRL/D to move the cursor down to the line below and delete the cursor character. Can be used with a repeat count to delete a column of spaces or characters in a table.

```
DEFINE KEY GOLD DELETE AS "DW."
```

Defines GOLD/DELETE to delete the entire word that the cursor is on.

```
DEFINE KEY GOLD CONTROL A AS "S/[name]/?'NAME: '/."
```

Defines GOLD/CTRL/A to search for the string **[name]** and then prompt the user to enter the name to be inserted at that place in the text.

7.3.5.4 Special Characters in Key Definitions — EDT uses several characters as signals in key definitions. These include the question mark (?), single quotations marks ('), and double quotation marks (").

Whenever you need to use a question mark literally in your key definition, you must enter two of them. If your key definition requires that EDT search for a question mark or insert one into your text, you must type two of them in your definition. The following command searches for a single question mark in your text:

```
DEFINE KEY GOLD Q AS "'??'."
```

When you use the SHOW KEY command, EDT displays both question marks.

```
SHOW KEY GOLD Q
'??'.
```

You only need to double quotation marks when your key definition is ambiguous. Ambiguity arises when you have used both single and double quotation marks and still need to reference quotation marks within the definition text. This occurs most often in a DEFINE KEY definition that uses the nokeypad "move" command to search for quotation marks.

```
DEFINE KEY GOLD B AS "'""' DC I^*^Z '""' DC I\*^Z."
```

In this definition, EDT first searches for a double quotation mark. It then deletes that character and inserts the RUNOFF bold flag (^*) in place of the quotation mark. Then EDT searches for a second quotation mark and replaces that with the closing RUNOFF bold flag (*). When you type the SHOW KEY command for this definition, EDT does not display the two double quotation marks that you entered.

```
SHOW KEY GOLD B
'"' DC I^*^Z '"' DC I\*^Z.
```

7.3.6 Key Definitions at Work

This section shows how to create some key definitions and use them to edit text.

Example 1: Transposing Letters

A common typographical error involves two letters in reverse order, for example, **teh** instead of **the**. You can define a key to transpose the two letters so that they are in the correct order. The definition is:

```
D+C +C UNDC
```

D + C deletes the character that the cursor is on and stores that character in the delete character buffer. + C moves the cursor one character to the right to be in position for the UNDC command. **UNDC** inserts the contents of the delete character buffer to the left of the cursor.

You might decide to choose GOLD/T to have the transposing function. The DEFINE KEY definition looks like this:

```
DEFINE KEY GOLD T AS "D+C +C UNDC."
```

The plus signs are used to ensure that the cursor will move exactly as planned regardless of EDT's current direction. Notice the spaces in the definition text. Remember that EDT allows spaces between nokeypad commands but not between the entity (or other specifier) and the command that it goes with. The spaces between individual commands are optional in the line mode string. When you create key definitions in keypad mode by pressing keypad keys, no spaces appear in the definitions.

To create the same definition in keypad mode, start by pressing CTRL/K.

```
Press the key you wish to define
```

```
Now enter the definition terminated by ENTER
```

```
D+C+CUNDC.
```

Now whenever you come across transposed characters in your text, you can press GOLD/T to reverse the order. Be sure that the cursor is positioned on the first transposed character.

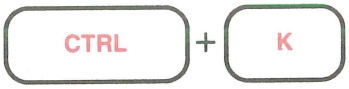
```
After reading the manu|a, I see how to do the task.
```

```
After reading the manua|, I see how to do the task.
```

Example 2: Relocating the HELP Key

Some EDT users prefer to relocate the HELP key so that it is not next to the GOLD key. This is especially useful if you are working at a slow terminal speed and do not want to take the time to wait for the HELP picture to be displayed when you accidentally press the wrong key. You might define GOLD/H to be the HELP key. You are advised not to use CTRL/H because this key is tied to the BACKSPACE key; by redefining CTRL/H, you also redefine BACKSPACE. But GOLD/H has no such restriction.

If you define GOLD/H in keypad mode before redefining the HELP key, you can press the HELP key to create the new definition.




```
Press the key you wish to define
```

```
Now enter the definition terminated by ENTER
```

```
HELP.
```

The preset HELP key is now available for redefinition.

You can also use the nokeypad HELP command to relocate the HELP function. In fact, the nokeypad HELP command exists solely for this purpose.

```
DEFINE KEY GOLD H AS "HELP."
```

If you want to make the change in a startup command file or EDT macro, you must type the nokeypad HELP command as your DEFINE KEY definition.

Example 3: Relocating CTRL/K

DEFK. is the definition for the CTRL/K function. Like the nokeypad HELP command, the nokeypad DEFK command exists only for the purpose of allowing you to relocate the CTRL/K function to another key or key sequence. Although DEFK and HELP are nokeypad commands, they perform no function in that mode. To relocate the key definition function to GOLD/ENTER with the DEFINE KEY command, type:

```
DEFINE KEY GOLD 21 AS "DEFK."
```

You can use CTRL/K to relocate the key definition function (providing you have not already assigned another function to that key sequence). However, you cannot press CTRL/K when EDT asks you to enter the definition, since that would insert a control K character (^K) into your definition. You must type **DEFK.** for the definition.

Example 4: Revising the SUBS Function

This example revises the substitute command so that you can enter the search and substitute strings directly from the terminal without having to load the search and PASTE buffers prior to pressing SUBS. The preset definition for SUBS is:

```
(CUTSR=DELETE PASTEKS"").
```

This definition relies on string-1 already being in the search buffer and string-2 being in the PASTE buffer. The default SUBS command also moves to the next occurrence of the search string after the substitution is made. You might find it easier to set up a substitute command that prompts you for both the search string and the substitute string when you press the key sequence. The following version of the command uses the substitute buffer, not the PASTE buffer:

```
DEFINE KEY GOLD 21 AS "S^@?'Find: '^@?' Substitute: '^@."
```

The **S** is the nokeypad substitute command. Instead of the slashes, which are often used as string delimiters, this example uses ^@ – the null character. By using the null character (decimal 0), you will be able to use a slash in either the search or substitute string without confusing EDT. Thus, you would be able to look for **I/O** and substitute **WORK I/O** for it. Of course, you cannot locate or substitute null characters with this version of the command, but you are more likely to be using the slash than the null character in a substitution.

If you plan to put such a DEFINE KEY command in a startup command file or an EDT macro, you must use a screen mode to insert the null characters (^@). You can use either the keypad SPECINS function or the nokeypad ASC command. If you are creating the definition interactively from keypad mode, you can use the CTRL/spacebar key sequence to insert the null character into the definition text.

The ending period means that the substitution will take place as soon as the strings have been entered. You do need to press the ENTER key after you type the search string and again when you type the substitute string. However, this gives you the chance to correct any typographical errors or change the string before EDT puts the data into the search and substitute buffers. Use the DELETE key to edit the strings as you type them.

The question marks signal EDT to accept input directly from the terminal. The first question mark is for the search string; the second for the substitute string. This example uses prompts inside the single quotation marks. **Find:** is the prompt for string-1; **Substitute:** the prompt for string-2. The spaces after the colons and before the word **Substitute** simply make the prompts more readable when they appear on the screen.

```
Please let me know by April 23 if you have any subbestions.
```

GOLD + ENTER SUBS

```
Find: subb
```

ENTER SUBS

```
Find: subb Substitute: sugg
```

ENTER SUBS

```
Please let me know by April 23 if you have any suggestions.
```

Example 5: Executing Nokeypad Commands Directly from Keypad Mode

There are times when you might want to use a nokeypad command without having to define a key to process that command. For example, you might want to test a key definition before assigning it to a key or key sequence. Or you might want to use a nokeypad entity such as PAR (paragraph) or a command such as DATE during your keypad session without having to define a key to do the work. This DEFINE KEY command defines CTRL/N to prompt you for a nokeypad command.

```
DEFINE KEY CONTROL N AS "?'Nokeypad Command: '."
```

The question mark signals EDT to accept input from the terminal. The prompt message is enclosed in single quotation marks. The space after the colon makes the prompt more readable when you are using it. The period at the end of the definition is the ENTER function that signals EDT to process the command. However, you must press ENTER to send the command statement to EDT. If you did not include the ENTER period, you would have to press ENTER twice – once to send the command to EDT, then again to have EDT process it.

```
A maximum distance of 800 feet which can be extended, is
```

CTRL + N

```
Nokeypad Command: -4DW
```

ENTER SUBS

```
A maximum distance of 800 feet is
```

Example 6: Line Mode Commands in Key Definitions

By using the nokeypad EXT (extend) command, you can define keys to process line mode commands. A number of line mode commands lend themselves to this type of usage. Earlier in this section in the discussion on using prompts with key definitions, you saw an example of the INCLUDE command and the WRITE command in key definitions.

When you use the nokeypad EXT command to access line mode commands, remember that you cannot use parentheses, repeat counts, or the GOLD/repeat feature. However, you can use the multicommand feature of line mode to include several line mode commands in your key definition. Also remember that spaces can be used wherever they are allowed in line mode command syntax.

The following sample DEFINE KEY commands access one or more line mode commands:

```
DEFINE KEY GOLD E AS "EXT EXIT."
```

Ends your EDT session directly from keypad mode. Copies the MAIN buffer text to an external file.

```
DEFINE KEY GOLD D AS "EXT DELETE REST."
```

Deletes all text from the current line through the end of the buffer.

```
DEFINE KEY GOLD C AS "EXT COPY SELECT TO =EXTRA."
```

Copies the select range to the buffer named EXTRA.

```
DEFINE KEY GOLD M AS "EXT MOVE =EXTRA TO ."
```

Deletes the text stored in the EXTRA buffer and inserts it above the current line.

```
DEFINE KEY GOLD S AS "EXT SUBSTITUTE+[module]+I/O+ . ."
```

Replaces the string **[module]** with the string **I/O**. Uses the + sign as the string delimiter because the slash is contained in the substitute string. Any nonalphanumeric character can be used as string delimiter with the line mode SUBSTITUTE command except % and _. The percent is not allowed because it is used as a signal for certain range specifiers when they are used with the <null> command. The underscore cannot be used because EDT will confuse it with the underscore allowed in buffer names. The first period is the line mode period range specifier. Using it ensures that all occurrences of the string [module] on the current line will be affected by the SUBSTITUTE command. The second period is the ENTER function.

```
DEFINE KEY GOLD O AS "EXT INSERT_V1."
```

Performs the macro INSERT_V1. The macro name includes an underscore, which is the only nonalphanumeric character that you can include in a buffer name.

```
DEFINE KEY GOLD X AS "EXT INCLUDE INDEX.MAC =INDEX; DEFINE MACRO INDEX; INDEX; FIND=MAIN. ."
```

This key definition creates and processes a macro that consists of key definitions. The macro text is stored in an external file called INDEX.MAC. The definition does the following: (1) inserts the text of the file INDEX.MAC into a buffer named INDEX; (2) moves to that buffer; (3) issues the DEFINE MACRO command; (4) issues the new INDEX command to process all the key definitions contained in the macro text; and (5) returns to the most recent current line in the MAIN buffer.

7.3.7 Creating a Table with User-Defined Keys

To create a table or chart, you need to be able to produce both vertical and horizontal lines. You also need to be able to add or delete spaces, thus moving columns of text to the right or left. And, you need to be able to add more columns to your table or chart. Key definitions can help you do all of these operations.

The sample table shown in this section consists of a list of employees, their employee numbers, office locations, and telephone extensions. The table heading is Documentation Writers . To create such a table, you need a command to draw horizontal lines and one to insert a line that consists of nothing but spaces. The line of spaces is necessary for adding vertical lines to the table.

Define GOLD/_ to draw a horizontal line of 60 underscores.

```
Documentation Writers
[EOB]
```

```
Press the key you wish to define
```

```
Now enter the definition terminated by ENTER
60(I_^Z).
```

Next, define a key to enter a line of 60 spaces.

```
Press the key you wish to define
```

```
Now enter the definition terminated by ENTER
60(I␣^Z).
```

Now you can start typing the table.

GOLD/_ + RETURN

GOLD/S + RETURN

```
Documentation Writers

------------------------------------------------------------
```

Note that the cursor is at the end of the line of spaces.

Type the headings on the next two lines, using GOLD/_ to draw the next horizontal line and GOLD/S to put a blank line after the horizontal line.

```
Documentation Writers

------------------------------------------------------------

Name                    Employee      Location      Telephone
                        Number                      Extension
------------------------------------------------------------
```

You are now ready to type the information into the table.

```
Documentation Writers

------------------------------------------------------------

Name                    Employee      Location      Telephone
                        Number                      Extension
------------------------------------------------------------

Baker, Kimberly         48609         PK2-7/73      4729
```

It occurs to you that the number of spaces between Employee Number and Location, as well as between Location and Telephone Extension, will always be the same. The first spacing has eight blanks; the second five. Define GOLD/L to be eight spaces and GOLD/T to be five spaces (L because you are moving to Location; T because you are moving to Telephone).

CTRL/K

```
Press the key you wish to define
```

GOLD + L

```
Now enter the definition terminated by ENTER
8(I␣^Z).
```

CTRL/K

```
Press the key you wish to define
```

GOLD + T

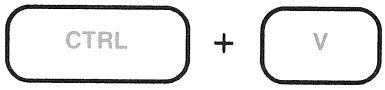
```
Now enter the definition terminated by ENTER
5(I␣^Z).
```

You can now type the remaining names in the list. Type in the name and then space over to enter the employee number. Next, use GOLD/L to space over to the Location column, and GOLD/T to move to the Telephone column.

```
Documentation Writers

------------------------------------------------------------

Name                    Employee      Location      Telephone
                        Number                      Extension
------------------------------------------------------------

Baker, Kimberly         48609         PK2-7/73      4729
Cole, Roger             34829         RT1-6/36      2984
Fenske, John            27482         LZ1-5/14      8463
Griffiths, Annabel      47659         GC2-2/12      9457
Hammel, Gary            17489         RT1-6/45      2784
   .
   .
   .
West, Anne Louise       46327         LZ1-3/54      3768
```

Once you have entered all the names, you are ready to draw the vertical lines. You could count up the exact number of lines in the table and use that number as the repeat count. But, a few extra key strokes are easier than figuring out exactly how many vertical bars are needed. By enclosing the definition that inserts a vertical bar in parentheses, you can use the keypad GOLD/repeat feature. Remember to place the period outside the parentheses.

```
Press the key you wish to define
```




```
Now enter the definition terminated by ENTER
(+VD-CI|^Z).
```

Position the cursor on the top horizontal line of the table on the 20th underscore. Now press GOLD/50 followed by CTRL/V to put 50 vertical bars in the table. Use CTRL/V with a repeat count as many times as you need to finish drawing the line. Then move the cursor to the location for the second vertical line and use CTRL/V with repeat counts to draw that line.

```
Documentation Writers

------------------------------------------------------------
                  |             |              |
Name              |    Employee |    Location  |   Telephone
                  |    Number   |              |   Extension
------------------ ------------- -------------- --------------
                  |             |              |
Baker, Kimberly   |    48609    |    PK2-7/73  |   4729
Cole, Roger       |    34829    |    RT1-6/36  |   2984
Fenske, John      |    27482    |    LZ1-5/14  |   8463
Griffiths, Annabel|    47659    |    GC2-2/12  |   9457
Hammel, Gary      |    17489    |    RT1-6/45  |   2784
  .
  .
  .
West, Anne Louise |    46327    |    LZ1-3/54     3768
```

When you look at the table, you decide that the vertical lines could be more evenly spaced. Again, you can define a key to add two spaces before the vertical bars. Enclose the definition in parentheses so it can be used with a repeat count.

CTRL/K

```
Press the key you wish to define
```

CTRL + P

```
Now enter the definition terminated by ENTER
(+VI▵▵^Z-2C).
```

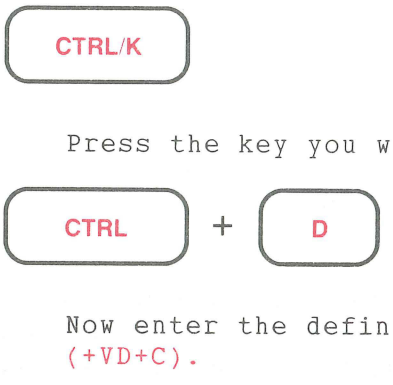
Position the cursor on the top horizontal line above the vertical line you want to move. Press GOLD/50 and then CTRL/P to move the vertical line. Then continue using GOLD/repeat with CTRL/P to move over the entire line. Repeat the steps until all three vertical lines in the table have been moved.

```
Documentation Writers

_____________________________________________________________
                    |                |                |
Name                |    Employee    |    Location    |   Telephone
                    |    Number      |                |   Extension
____________________|________________|________________|_______________
                    |                |                |
Baker, Kimberly     |    48609       |    PK2-7/73    |    4729
Cole, Roger         |    34829       |    RT1-6/36    |    2984
Fenske, John        |    27482       |    LZ1-5/14    |    8463
Griffiths, Annabel  |    47659       |    GC2-2/12    |    9457
Hammel, Gary        |    17489       |    RT1-6/45    |    2784
  .
  .
  .
West, Anne Louise   |    46327       |    LZ1-3/54    |    3768
```

The horizontal lines need some minor repairs. The top one needs to have six more underscores added to the end of it and the other two need the spaces replaced by underscores. But this is a small price to pay for the convenience of being able to move the lines over with just a few keystrokes.

```
Documentation Writers

_____________________________________________________________
                    |                |                |
Name                |    Employee    |    Location    |   Telephone
                    |    Number      |                |   Extension
____________________|________________|________________|_______________
                    |                |                |
Baker, Kimberly     |    48609       |    PK2-7/73    |    4729
Cole, Roger         |    34829       |    RT1-6/36    |    2984
Fenske, John        |    27482       |    LZ1-5/14    |    8463
Griffiths, Annabel  |    47659       |    GC2-2/12    |    9457
Hammel, Gary        |    17489       |    RT1-6/45    |    2784
  .
  .
  .
West, Anne Louise   |    46327       |    LZ1-3/54    |    3768
```

Another key definition that you will find useful when creating tables deletes a vertical row of spaces so that you can bring columns closer together. Define CTRL/D to delete space from the current line and move the cursor to the same column on the next line.



The next section, Defining EDT Macros, shows how to construct a macro that includes all of these key definitions, except those that were specific to this example. You can store the commands in an external file. Whenever you want to use the functions to format a table or chart, simply use the line mode INCLUDE command to put them into the macro buffer for use during your editing session.

7.4 Defining EDT Macros

An EDT macro is a group of instructions that is put into effect when you type the macro name as a line mode command. In EDT you use the DEFINE MACRO command to add the macro name to the list of valid line mode commands for the duration of your editing session. The contents of the macro must be placed in a separate EDT buffer with the same name as that used with the DEFINE MACRO command. When you want EDT to process the macro, type the macro name just as you would any other line mode command.

Although macros can contain only line mode commands, they are able to perform a variety of functions. Often macros contain SET commands to customize your editing session. The line mode DEFINE KEY command enables you to include key definitions in a macro. A macro can be used to define other macros or to call up macros from external files. Two or more line mode commands such as FILL and RESEQUENCE can be combined into one command. For example, you could create a macro called FILLSEQ that causes EDT to reformat an entire buffer and then resequence it.

EDT macros can do the same things that startup command files do. Macros have the added flexibility that you do not need to stop your EDT session in order to change from one group of commands to another. For example, you can use macros to switch back and forth between certain SET commands and their defaults. By creating two macros – one with the default SET commands and the other with the alternate group – you can easily access whichever group you want by typing its macro name.

You can create macros any time during your editing session and then use them for the remainder of the session. Since macros are stored in buffers, they disappear as soon as you end your editing work. If you develop a macro that you want to keep for other EDT sessions, you can use

the WRITE command to put a copy of the macro in an external file. When you need that macro again, use the INCLUDE command to copy the macro into a buffer and then establish the macro name as a command with the DEFINE MACRO command.

The syntax for EDT's macro command is:

```
DEFINE MACRO macro-name
```

Macro names can be as long as buffer names. However, if you expect to type the macro name several times during your session, you will want to keep the name short. If you plan to save your macros, you might want to limit the name to the number of characters allowed for file names by your operating system. When writing a macro to an external file, you can use the file type **.MC** or **.MAC** to call attention to the fact that the file contains a macro.

To use EDT's macro facility:

1. Issue the DEFINE MACRO command.
2. Insert the macro text into a buffer with the same name as the macro name given with the DEFINE MACRO command.

It does not matter in which order you carry out these steps. Once you issue the DEFINE MACRO command, EDT adds the macro name to its list of valid line mode commands. If you issue the DEFINE MACRO command first, EDT creates a buffer with the name you supplied for **macro-name**, but does not move to that buffer.

It is possible to have a macro name be the same as an existing line mode command. When this happens, EDT performs only the macro whenever the command name is typed. The default command has been superseded and no longer exists as long as the macro is in effect. If you need to re-establish the default use of that command, use the CLEAR command to eliminate the buffer with that macro name. EDT then resumes using the default function for that command name.

When using line mode commands that take specifiers, you must include the appropriate specifier in the macro text. For example, if you create a macro that includes a substitute command, you must supply the strings for EDT to use with that substitute command – SUBSTITUTE/minicom/microcom/. You cannot designate the macro to be the SUBSTITUTE command and expect to enter the strings after typing the macro name.

The same restriction applies when you use a command like TYPE in your macro. If you want to have a command that automatically types the current line and the next 10 lines, you can create the following macro:

```
TYPE . THRU +9
```

You cannot, however, create a macro that will allow you to enter the number of lines for EDT to display whenever you type the macro name.

If you are creating a macro that inserts text, use the single line form of the INSERT command: the command word INSERT, then a semicolon, and finally the line of text you want to insert.

You can revise or edit your macro any time during your editing session. Move to the macro buffer and use any EDT commands or functions to perform the necessary changes, additions, or deletions. Now you are ready to use the same macro name to produce the new operations.

Here are some examples of macros you might create and the processes used to create them.

Example 1: Alternating Search Parameters

This example uses line mode to create two search macros: one with SET SEARCH EXACT and SET SEARCH END and the other with the default values. With these two macros in place you can change the search parameters any time you want by typing the appropriate macro name.

```
*DEFINE MACRO EXACT
*FIND =EXACT
*INSERT
          SET SEARCH EXACT
          SET SEARCH END
          ^Z
[EOB]
*DEFINE MACRO GENERAL
*FIND =GENERAL
*INSERT
          SET SEARCH GENERAL
          SET SEARCH BEGIN
          ^Z
[EOB]
*FIND =MAIN.
```

Now that you have both macros in place, you are ready to begin your work. You can switch back and forth between the two different kinds of search parameters by issuing the appropriate macro name as a line mode command.

Example 2: Displaying a Group of Lines in Line Mode

Suppose you are working in line mode. You want a single command that causes EDT to display a group of lines and then leave the cursor at the beginning of the last line in the group, instead of stopping at the first line, which is the normal case. Creating the macro called TYPE10 could be done this way:

```
*DEFINE MACRO TYPE10
*FIND =TYPE10
*INSERT
          TYPE . THRU +9
          FIND +9
          ^Z
*FIND =MAIN
```

Next time you want to see 10 lines of text displayed, type the macro name after the asterisk prompt (*).

```
   26         however, when the agenda was established, no one
*TYPE10
   26         however, when the agenda was established, no one
   27         who attended that meeting had any objections.
   28         Now that we have been meeting biweekly, the
   29         attendance has dropped sharply.  In fact, the
   30         last meeting had to be cancelled because only four
   31         people showed up.
```

```
   32
   33          These meetings are supposed to benefit the entire
   34          staff, not just a few individuals.  If people
   35          feel that the meetings are a waste of time, I
*+1
   36          can cancel the remaining ones.
```

Example 3: Inserting a Group of Lines

You can use a macro to insert more than one line by supplying a separate INSERT command for each line. This example inserts the closing for a letter:

```
*DEFINE MACRO CLOSING
*FIND =CLOSING
*INSERT
          INSERT;Sincerely yours,
          INSERT;
          INSERT;
          INSERT;
          INSERT;
          INSERT;George C. Martingale
          INSERT;President
          INSERT;The Society for the Preservation of
          INSERT;  Purple Martins and Nightingales
          INSERT;Suite 3500
          INSERT;4869 Avenue of the Americas
          INSERT;New York, New York  10058
          ^Z
*FIND =MAIN.
```

When you finish typing your letter, give the newly created CLOSING command and EDT will insert the entire closing from either line mode or a screen mode.

```
look forward to seeing you in the coming weeks.

```

GOLD + PAGE COMMAND

```
Command: CLOSING
```

ENTER SUBS

```
          Sincerely yours,



          George C. Martingale
          President
          The Society for the Preservation of
            Purple Martins and Nightingales
          Suite 3500
          4869 Avenue of the Americas
          New York, New York  10058
```

Example 4: Creating the TABLE Macro

The section Creating a Table with User-Defined Keys, which appears earlier in this chapter, shows how to create key definitions that can help you to format a table or chart. You can create a macro that contains the various key definitions that were used in that application. Of course, you will need to use the DEFINE KEY command to create your key definitions since macros can only contain line mode commands. Once you have entered all the definitions in the macro buffer, you can store the macro in an external file for use in later editing sessions.

```
*DEFINE MACRO TABLE
*FIND =TABLE
*INSERT
          DEFINE KEY GOLD _ AS "60(I_^Z)."
          DEFINE KEY GOLD S AS "60(I^Z)."
          DEFINE KEY CONTROL V AS "(+V D-C I|^Z)."
          DEFINE KEY CONTROL P AS "(+V I^Z -C)."
          DEFINE KEY CONTROL D AS "(+V D+C)."
          ^Z
*WRITE TABLE.MAC
*FIND =MAIN.
*TABLE
```

The first command defines the macro to be TABLE. Then EDT enters the TABLE buffer, so you can insert the key definitions for the macro. The key definitions are the same as those described in the section on creating a table, earlier in this chapter.

```
DEFINE KEY GOLD _ AS "60(I_^Z)."
```
Inserts 60 underscore characters on the current line.

```
DEFINE KEY GOLD S AS "60(I^Z)."
```
Inserts 60 spaces on the current line.

```
DEFINE KEY CONTROL V AS "(+V D-C I|^Z)."
```
Draws a vertical line, one vertical bar at a time.
Parentheses allow you to use a repeat count.

```
DEFINE KEY CONTROL P AS "(+V I^Z -C)."
```
Adds a column of spaces, one space at a time.
Parentheses allow you to use a repeat count.

```
DEFINE KEY CONTROL D AS "(+V D+C)."
```
Removes a column of spaces, one space at a time.
Parentheses allow you to use a repeat count.

These five key definitions form the core of the functions you need to format a table or chart. When you finish typing them in the TABLE buffer, use the WRITE command to store them in an external file for future use. Then return to the MAIN buffer and issue the new TABLE command to have EDT process the five DEFINE KEY commands. The new editing keys are now ready for use.

During the course of creating your table, you might want to add more key definitions to your editing session. If you are sure that you will not need these again, you need not worry about adding them to the TABLE buffer. But if you find you cannot finish the table in one editing session, you can add the new definitions to the TABLE buffer and put the contents in another external file. In the section on creating a table, there were two additional commands: one moved the cursor from column 2 to column 3; the other moved the cursor from column 3 to column 4.

To add these commands to the TABLE macro, move to the TABLE buffer and type the new definitions at the end of the buffer. Then, because you know you want to use the entire macro later on, you can put the contents of the TABLE buffer in an external file called TABLE1.MAC. Now return to your place in the MAIN buffer to continue working on the table. Issue the TABLE command again to be sure that EDT processes the additional key definitions for GOLD/L and GOLD/T.

```
*FIND =TABLE END
*INSERT
          DEFINE KEY GOLD L AS "8(I^Z)."
          DEFINE KEY GOLD T AS "5(I^Z)."
          ^Z
*WRITE TABLE1.MAC
*TABLE
*FIND =MAIN.
```

When you are ready to resume work on the table, simply use the INCLUDE command to put the macro in a buffer called TABLE, define the macro for the new editing session, issue the TABLE command to set up the key definitions, and return to the MAIN buffer.

```
*INCLUDE TABLE1.MAC =TABLE
*DEFINE MACRO TABLE
*TABLE
*FIND =MAIN.
```

Now all the key definitions are in place.

Example 5: Macros in Startup Command Files

You can include the text of a macro and all the commands necessary to process that macro in a startup command file. It is also possible to define a key to load a macro that is stored in an external file and then process the necessary macro commands.

The first command to create the macro is the FIND command, which moves EDT to the macro buffer.

```
FIND =TABLE
```

In order to include the text of a macro in your startup command file, you must set up each line of the macro text as a single-line line mode INSERT command.

```
INSERT; DEFINE KEY GOLD _ AS "60(I_^Z)."
INSERT; DEFINE KEY GOLD S AS "60(I^Z)."
INSERT; DEFINE KEY CONTROL V AS "(+V D-C I|^Z)."
INSERT; DEFINE KEY CONTROL P AS "(+V I^Z -C)."
INSERT; DEFINE KEY CONTROL D AS "(+V D+C)."
```

Now add the DEFINE MACRO command.

```
DEFINE MACRO TABLE
```

If your macro is composed of DEFINE KEY commands as this one is, you will want to issue the macro name immediately so that the key definitions are in place at the start of your editing session.

```
TABLE
```

Finally, you must add a second FIND command to return to the MAIN buffer for the start of your session.

```
FIND =MAIN
```

The portion of your startup command file that is devoted to the macro looks like this:

```
FIND =TABLE
INSERT; DEFINE KEY GOLD _ AS "␆␀(I_^Z)."
INSERT; DEFINE KEY GOLD S AS "␆␀(I␣^Z)."
INSERT; DEFINE KEY CONTROL V AS "(+V D-C I|^Z)."
INSERT; DEFINE KEY CONTROL P AS "(+V I␣^Z -C)."
INSERT; DEFINE KEY CONTROL D AS "(+V D+C)."
DEFINE MACRO TABLE
TABLE
FIND =MAIN
```

If you have the macro text stored in an external file, you can replace all the single-line INSERT commands with an INCLUDE command.

```
FIND =TABLE
INCLUDE TABLE.MAC
DEFINE MACRO TABLE
TABLE
FIND =MAIN
```

If you do not plan to use the macro all the time, you can define a key to process all the commands needed to set up the macro. Be sure to keep the definition on a single line.

```
DEFINE KEY GOLD T AS "EXT FIND =TABLE ;INCLUDE TABLE.MAC ;DEFINE MACRO TABLE ;TABLE ;FIND =MAIN. ."
```

Note the space between the file specification TABLE.MAC and the semicolon (;). Although it is not generally necessary to have a space before the semicolon separating two line mode commands, in this case omitting the space causes a problem for file specifications on systems that use the semicolon to separate the file version number from the file type. If you issue this EDT command on such an operating system and omit that space, EDT becomes confused about what file to include and generally tells you that it cannot find the file. You can avoid this problem by using the buffer specifier in the INCLUDE command and omitting the initial FIND =TABLE command. The definition looks like this:

```
DEFINE KEY GOLD T AS "EXT INCLUDE TABLE.MAC =TABLE ;DEFINE MACRO TABLE ;TABLE ;FIND =MAIN. ."
```

In this definition, the space after =TABLE is optional.

7.5 Shifting Editing Modes

EDT has several ways to go from one editing mode to another. In some cases, EDT makes a total shift from one mode to another. In other cases, the shift in mode lasts only for one command line. Not all shifts can be accomplished from each mode.

7.5.1 Moving from Line Mode to the Change Modes

When you are in line mode, you will generally be making a complete shift to one of the change modes: keypad, nokeypad, or hardcopy change mode. (If you are using a terminal that does not support the screen modes, you can only shift to hardcopy change mode.) However, there is a way to issue a nokeypad command line from line mode.

7.5.1.1 Making a Complete Shift from Line Mode to a Change Mode — The CHANGE command makes the shift from line mode to a screen mode, as well as to hardcopy change mode. The SET [NO]KEYPAD command determines which screen mode – keypad or nokeypad. EDT's default setting is KEYPAD. Thus, when you start your EDT session, you only need to type CHANGE to shift to keypad mode.

```
*CHANGE
```

To enter nokeypad mode, you must first issue the SET NOKEYPAD command, then type CHANGE.

```
*SET NOKEYPAD
*CHANGE
```

Once SET NOKEYPAD is in effect, you must issue the SET KEYPAD command to shift to keypad editing.

If your terminal is not supported for screen mode editing, you still have the option of using hardcopy change mode. This editing mode enables you to use most nokeypad commands. A section describing hardcopy change mode appears later in this chapter. The example below shows how to enter that mode.

The CHANGE command shifts EDT to hardcopy change mode if your terminal is set to HCPY (hardcopy). You can use the SHOW TERMINAL command to check your terminal setting. Once you type the CHANGE command, EDT displays the current line and then the hardcopy change mode prompt – **C***. Whenever you see that prompt, you can enter most nokeypad commands for EDT to process.

```
*TYPE .
   10        This is the current line.
*CHANGE
[T]his is the current line.
C*
```

Notice the square brackets surrounding the **T** at the beginning of the first line. These brackets indicate the cursor position in hardcopy change mode.

The CHANGE command can take the line mode buffer and range specifiers as well as a nokeypad command line. The full syntax for CHANGE is:

```
CHANGE [=buffer] [range] [;nokeypad-command(s)]
```

You can supply the nokeypad command line even when you use the CHANGE command to shift to keypad mode.

```
CHANGE =CLOSING "cc:" ;EL IJames Fowler^Z
```

This command shifts to keypad mode, enters the CLOSING buffer, and moves to the line containing the string **cc:**. The nokeypad commands that follow the semicolon instruct EDT to move to the end of the line and insert the string **James Fowler**.

7.5.1.2 Using a Nokeypad Command Line from Line Mode — One of the optional specifiers with the CHANGE command is the **;nokeypad-command(s)** specifier. You can use this specifier with the CHANGE command whenever you are making a complete shift to a change mode. However, you can use the nokeypad EX (exit to line mode) command as the last element of the **;nokeypad-command(s)** specifier, to have EDT return to line mode after processing the nokeypad commands.

If your terminal does not support screen editing, you can use this feature of the CHANGE command to access an occasional nokeypad command instead of shifting to hardcopy change mode. For example, you could create a select range of a paragraph or page for use with the WRITE command:

```
*CHANGE ;SEL PAR EX
*WRITE INTROPAR.DAT SELECT
*CHANGE ;SEL PAGE EX
*WRITE PAGE2.RNO SELECT
```

You can use the CHANGE command to insert control characters, such as the form feed (CTRL/L) into your text.

```
     July 10, 1985

*CHANGE ;12ASC EX

     <FF>July 10, 1985
```

7.5.2 Shifting from Keypad Mode to Line or Nokeypad Mode

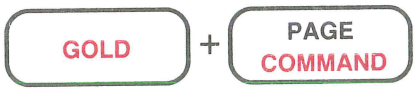
When you are in keypad mode, you can shift to either line mode or nokeypad mode. You can also access a single line mode command line or nokeypad command line any time during your keypad editing session without having to leave keypad mode.

7.5.2.1 Keypad to Line Mode — CTRL/Z shifts EDT from keypad mode into line mode. When you press CTRL/Z, EDT immediately displays the asterisk prompt (*) at the bottom left corner of the screen, indicating that you can issue line mode commands. You can issue as many line mode commands as you like. To reenter keypad mode, use the CHANGE command.

This example enters line mode to delete the remainder of the current buffer and then ends the editing session.

```
*DELETE REST
25 lines deleted
*EXIT
DISK$USER:[DUNSTER]MEMO.RNO  43 lines
```

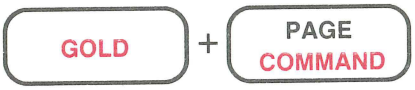
The keypad mode COMMAND function allows you to enter a line mode command any time during keypad editing. This function requires pressing the GOLD key and then the PAGE/COMMAND keypad editing key. When EDT displays the prompt **Command:** at the bottom of the screen, type the line mode command. Then press ENTER to have EDT process the command. (If you press RETURN, EDT displays ^M. Simply press the DELETE key once to remove the control symbol and then press ENTER to send the command to EDT.) For example, suppose you want to refresh your memory about the paragraph entity:




```
Command:  SHOW ENTITY PARAGRAPH
```

```
Command:  SHOW ENTITY PARAGRAPH
<CR><CR>
```

If you make a mistake in the line mode command, EDT displays the message indicating where the problem is on the command line. When you press RETURN to continue, EDT restores the cursor to your text. You must press GOLD/COMMAND to reissue the command.




```
Command:  COPY WHOLE TO EXTRA
```

```
Command:  COPY WHOLE TO EXTRA
                    ^
Unexpected characters after end of command
Press return to continue
```

In this case, EDT was unable to recognize EXTRA as the buffer name because it had neither the equal sign nor the BUFFER signal preceding it.

EDT can accept several line mode commands on a single line if the commands are separated with semicolons (;). Using the multicommand line feature, you can issue two or more line mode commands with a single use of the COMMAND function. When EDT displays the **Command:** prompt, type as many line mode commands as you need. The total number of characters that EDT can accept on a single command line is 255, but EDT can only display the characters up to the right margin of the screen, so you probably will not want to type more than two or three commands at once.

You can use the multicommand line feature to process several SHOW commands at one time.

GOLD + PAGE COMMAND

```
Command: SHOW SCREEN ; SHOW LINES ; SHOW CURSOR ; SHOW WRAP
```

ENTER SUBS

```
Command: SHOW SCREEN ; SHOW LINES ; SHOW CURSOR ; SHOW WRAP
80
22
7:14
nowrap
Press return to continue
```

You might want to use COPY to put text in a buffer and then return to the MAIN buffer in the same command line.

```
Command: COPY SELECT TO =EXTRA ; FIND =MAIN.
```

7.5.2.2 Keypad to Nokeypad — There are two ways to shift from keypad mode to nokeypad. Using the first way, you exit from keypad mode and then issue the SET NOKEYPAD and CHANGE commands. The other way uses the keypad mode COMMAND function. The shift to nokeypad mode by way of line mode looks like this:

CTRL/Z

```
*SET NOKEYPAD
*CHANGE
```

EDT is now set for nokeypad screen mode. To return to keypad editing you must issue the SET KEYPAD command. You can use the buffer and range specifiers as well as the **;nokeypad-command(s)** specifier with the line mode CHANGE command.

The keypad COMMAND function enables you to go directly from keypad mode to nokeypad.

```
Command:  SET NOKEYPAD
```

Another way to access nokeypad commands from keypad mode uses a key definition that prompts you for a single nokeypad command line. You can use CTRL/K to define the key in keypad mode. If you expect to use this feature in many editing sessions, put a DEFINE KEY command with the definition in a startup command file or in an EDT macro. (Information on EDT's key definition facility and on startup command files is given in other sections of this chapter.)

The syntax for the key definition is:

```
?['prompt text'].
```

The prompt is optional, but it is helpful to have EDT display some text so that you know which kind of information to supply. Any wording is possible, for example:

```
Nokeypad Command:

Nokeypad Here --.
```

This example uses CTRL/N as the key sequence to press when you want to enter a nokeypad command.

```
Press the key you wish to define
```

```
Now enter the definition terminated by ENTER

?'Nokeypad Command: '.
```

If you include a space after the prompt text, it is easier to see the command as you are typing it.

The line mode version for this key definition is:

```
DEFINE KEY CONTROL N AS "?'Nokeypad Command: '."
```

To use this definition, first press the appropriate key sequence. As soon as EDT displays the prompt, type the nokeypad command you want EDT to process. You can use the nokeypad multicommand feature to have EDT process several commands. Press ENTER to send the nokeypad command(s) to EDT for processing.

The following example shows how the newly defined key sequence works:

```
We are going to delete three words to the left of
the cursor.  One!  Two!  Three!
```

CTRL/N

```
Nokeypad Command: -D3W
```

ENTER SUBS

```
We are going to delete three words to the left of
the cursor.
```

7.5.3 Shifting from Nokeypad Mode to Keypad Mode or Line Mode

EDT has ways for you to shift from nokeypad to line mode completely or for a single line mode command line. You can also shift completely to keypad mode. However, there is no way to have EDT perform a single keypad operation while you are in nokeypad mode.

7.5.3.1 Nokeypad to Line Mode — When you are working in nokeypad mode you have the choice of making a shift to line mode or accessing line mode for a single command line.

Nokeypad editing uses the EX (exit to line mode) command to shift to line mode. Simply type EX; the line mode asterisk prompt (*) appears as soon as you press RETURN.

```
EX
*
```

To use line mode commands while still editing with nokeypad, use the nokeypad EXT (extend) command. First type EXT, next type a line mode command, and then press RETURN.

```
EXT COPY . THRU END TO =CODEX
```

You can use two or more line mode commands with EXT by separating the line mode commands with semicolons (;).

```
EXT SET WRAP 50; SET LINES 10; SET CURSOR 3:7
```

If you want to add nokeypad commands after the last line mode command, add the CHANGE command followed by a semicolon and the nokeypad commands.

```
EXT SET QUIET; SET TAB 5; CHANGE ;TADJSR PAGE
```

7.5.3.2 Nokeypad to Keypad — The shift from nokeypad editing to keypad mode uses the nokeypad EXT command to issue the line mode SET KEYPAD command.

```
EXT SET KEYPAD
```

As soon as EDT adjusts the screen, you are ready to press the keypad keys to have EDT perform keypad editing functions.

7.6 The EDT Journal File

In Chapter 2 you saw how to use the /RECOVER qualifier with your EDIT/EDT command line to run EDT's journal facility. This section shows you what EDT journal files look like and how to delete an unwanted command from the end of a journal file.

7.6.1 Looking at the Journal File

EDT uses the journal file to record every keystroke you make during your EDT session. Because of the different natures of line mode and keypad mode, the journal file from a line editing session looks distinctly different from that of a keypad session. The line mode journal file is straightforward. It is easy to follow the commands and insertions you made during your editing session. In keypad mode, however, the journal file contains many escape character symbols that correspond to the editing keys that you press. These escape sequences require some interpretation to determine which keystrokes you made.

This section shows two examples of journal files: the first one is a line mode journal file; the second one is a keypad mode journal file. Both sample sessions use the **QUIT/SAVE** command to save a copy of the journal file.

The line mode example starts by entering the sample text into the empty MAIN buffer, then makes some changes in the text, and finally uses **QUIT/SAVE** to simulate a system interruption.

```
☞ EDIT /EDT 16JUN85.DAT
Input file does not exist
[EOB]
*INSERT
         9:00     Meeting of Board of Directors
        12:00     Lunch with Mr. Peters of Federal Energy Co.
         2:30     Meeting with Mr. Jones of Advertising
         5:45     Taxi to airport
         6:37     Flight AA578 to Phoenix
                       Hotel Reservations at Ramada Inn
           ^Z
[EOB]
*TYPE WHOLE
   .
   .
   .
```

Now, before exiting from EDT, you find that you need to make some changes in the schedule. Mr. Peters has been replaced by Mr. Rafferty, the flight time and flight number are changed, and the hotel reservations have been switched. As soon as you finish typing all the changes, a system interruption, which you simulate with the QUIT/SAVE command, occurs.

```
*SUBSTITUTE/Peters/Rafferty/2
    2       12:00     Lunch with Mr. Rafferty of Federal Energy Co.
1 substitution
*SUBSTITUTE/37/54/5
    5        6:54     Flight AA578 to Phoenix
1 substitution
*SUBSTITUTE/AA578/TWA235/
    5        6:54     Flight TWA235 to Phoenix
1 substitution
*SUBSTITUTE/Ramada Inn/Best Western/6
    6                      Hotel Reservations at Best Western
1 substitution
*QUIT/SAVE
```

The journal file looks like this:

```
INSERT
 9:00     Meeting of Board of Directors
12:00     Lunch with Mr. Peters of Federal Energy Co.
 2:30     Meeting with Mr. Jones of Advertising
 5:45     Taxi to airport
 6:37     Flight AA578 to Phoenix
               Hotel reservations at Ramada Inn
^Z
TYPE WHOLE
SUBSTITUTE/Peters/Rafferty/2
SUBSTITUTE/37/54/5
SUBSTITUTE/AA578/TWA235/
SUBSTITUTE/Ramada Inn/Best Western/6
QUIT/SAVE
```

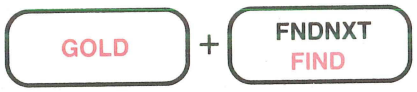
After using the /RECOVER qualifier with the EDIT/EDT command line, EDT restores the editing session to the point just before the QUIT/SAVE command. If you try to recover your EDT session after a system interruption, EDT does not always process the last few commands. This is because EDT saves the journal entries in blocks of several operations at a time. A system interruption does not always occur at the start of a new journal file block. If the block has not yet been transferred to the journal file when the interruption occurs, those commands currently stored in that block will be lost. However, no other commands from earlier in your editing session will be missing.

The TYPE WHOLE command shows the text after the recovery:

```
*TYPE WHOLE
    1       9:00     Meeting of Board of Directors
    2      12:00     Lunch with Mr. Rafferty of Federal Energy Co.
    3       2:30     Meeting with Mr. Jones of Advertising
    4       5:45     Taxi to airport
    5       6:54     Flight TWA235 to Phoenix
    6                     Hotel reservations at Best Western
[EOB]
```

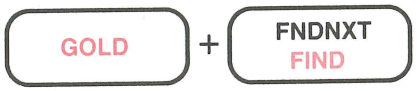
When you use the journal facility in keypad mode, you get the same results, but the journal file looks very different from the line mode version. The next example assumes that you typed the first version of the schedule and used the EXIT command to save a copy. Now make the changes in keypad mode. Use QUIT/SAVE to simulate the system interruption once the changes have been made.

```
☞ EDIT/EDT 16JUN85.DAT
    1       9:00      Meeting of Board of Directors
*CHANGE



 9:00      Meeting of Board of Directors
12:00      Lunch with Mr. Peters of Federal Energy Co.
 2:30      Meeting with Mr. Jones of Advertising
 5:45      Taxi to airport
 6:37      Flight AA578 to Phoenix
               Hotel Reservations at Ramada Inn
```

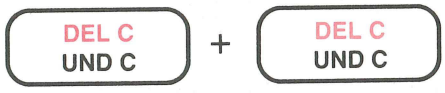



```
Search for: Peters
```

```
Rafferty▵
```




```
Search for: 37
```



54

GOLD + FNDNXT FIND

Search for: AA578

ENTER SUBS

DEL W UND W

TWA235

GOLD + FNDNXT FIND

Search for: Ramada Inn

ENTER SUBS

DEL W UND W + DEL W UND W

Best Western

GOLD + PAGE COMMAND

Command: QUIT/SAVE

ENTER SUBS

Here is the way the journal file looks:

```
CHANGE
<ESC>OP<ESC>ORPeters<ESC>OM<ESC>OmRafferty <ESC>OP<ESC>OR3?<ESC>OM
<ESC>Ol<ESC>O154<ESC>OP<ESC>ORAA578<ESC>OM
<ESC>OmTWA235 <ESC>OP<ESC>ORRamada Inn<ESC>OM
<ESC>Om<ESC>OmBest Western<ESC>OP<ESC>OwQUIT/SAVE<ESC>OM
```

One of the obvious differences between the keypad journal file and the line mode one is the escape characters (<ESC>) that appear throughout the file. <ESC> and the following letter(s) are signals sent to EDT when keypad editing keys are pressed. For example, <ESC>OP corresponds to pressing GOLD; <ESC>OR to the FNDNXT/FIND key.

7.6.2 Editing the Journal File

There are times when it is useful to be able to edit the journal file. The only editing operation you should perform on a journal file is to delete from a particular point to the end. In a line mode journal file, it is easy to find the deletion point. But you can get confused when you try to find the right spot in the keypad version.

The most common reason for editing a journal file is to remove a DELETE command in line mode or a CUT command in a screen mode. When you find that you have just removed a large chunk of text by mistake and that the only record of that text exists in the original file or as inserted text in the journal file, it is time to resort to the QUIT/SAVE command. QUIT/SAVE ends your EDT session without saving a copy of the MAIN buffer, but does save a copy of the journal file.

If you are creating a new file with EDT, no copy exists in any directory. But the information stored in the journal file enables EDT to restore the text, nonetheless.

Suppose in the previous sample text, you wanted to delete the last line – the one about the hotel reservations. But instead of typing DELETE 6, you typed DELETE WHOLE.

```
   .
   .
   .
*DELETE WHOLE
6 lines deleted
[EOB]
*QUIT/SAVE
☞
```

Using EDT to edit the journal file, you can remove the unwanted DELETE command. Note the /NOJOURNAL qualifier in the EDIT/EDT command line in the next sample session. The /NOJOURNAL qualifier prevents EDT from writing more data to the journal file called 16JUN85.JOU. If a system interruption occurred while you were editing the journal file, you would be unable to use that file to recover either editing session.

Also notice the DELETE command that you used to remove the unwanted DELETE WHOLE line. This command calls for EDT to delete the remainder of the text – from line 13 to the end. To avoid confusing EDT, you should always delete everything from the command you want to eliminate through the end of the journal text. Otherwise, EDT can produce a disorganized file or make edits in the wrong places.

```
☞ EDIT /EDT /NOJOURNAL 16JUN85.JOU
    1       INSERT
*TYPE WHOLE

    1       INSERT
    2       9:00     Meeting with Board of Directors
    .
    .
    .
   13       DELETE WHOLE
[EOB]
*DELETE 13 THRU END
2 lines deleted
[EOB]
*EXIT
```

Now you are ready to use the recovery facility:

```
☞ EDIT /EDT /RECOVER 16JUN85.DAT
```

This is what you see:

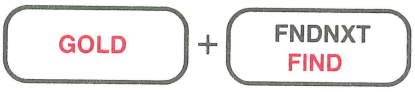
```
Input file does not exist
INSERT
 9:00      Meeting of Board of Directors
12:00      Lunch with Mr. Peters of Federal Energy Co.
 2:30      Meeting with Mr. Jones of Advertising
 5:45      Taxi to airport
 6:37      Flight AA578 to Phoenix
           Hotel reservations at Ramada Inn
^Z
SUBSTITUTE/Peters/Rafferty/2
    2          12:00      Lunch with Mr. Rafferty of Federal Energy Co.
SUBSTITUTE/37/485
    5           6:48      Flight AA578 to Phoenix
SUBSTITUTE/AA578/TWA235/
    5           6:48      Flight TWA235 to Phoenix
SUBSTITUTE/Ramada Inn/Best Western/6
    6                     Hotel reservations at Best Western
*
```

Editing a keypad journal file can be considerably more difficult than the line mode version, but not impossible. In keypad editing, EDT records the transmission codes for the editing keys you press. The user's guide for your terminal lists the codes transmitted by each key.

Generally you will want to edit a journal file because you have used CUT to remove text from a buffer and then used another CUT command, which overwrites the contents of the PASTE buffer. To restore the missing text, you need to search for the appropriate CUT escape sequence. Most of the time, the CUT escape sequence will follow a SELECT escape sequence. Between the SELECT and CUT sequences come the cursor moving functions that you use to create the select range. These will give you a clue to which CUT operation you are seeing. For example, in this piece of the journal file, two lines were moved to the PASTE buffer:

```
□<ESC>[A<ESC>[A<ESC>[A<ESC>On<ESC>Op<ESC>Op<ESC>Ov
```

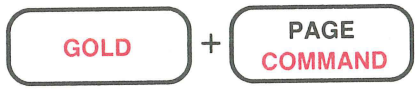
To edit the journal file containing this line, move the cursor to the **<ESC>On** sequence. Then delete everything from the cursor position to the end of the text.




```
Search for: <ESC>On
```

```
<ESC>[A<ESC>[A<ESC>[A□<ESC>On<ESC>Op<ESC>Op<ESC>Ov
```

Now move the cursor to the beginning of the next line and delete the remaining text in the buffer.




```
Command: DELETE REST
```

```
Command: DELETE REST
5 lines deleted
```

Occasionally, when you are editing a file, you might find that the same edits you did at the beginning of the session are applicable to another file you need to edit. You can use EXIT/SAVE at the end of the session to save both the edited version of the text and the journal file. Find the point in the journal file where the last edit common to both files occurs. Then delete the remaining material from the journal file. Use the /RECOVER qualifier and the /JOURNAL qualifier with the journal file name when you are ready to edit the other file. In order to use this procedure, however, you must be absolutely sure that the starting files are identical and that all other elements of the editing session will be the same up to the point where EDT completes processing the journal file.

The next sample session shows the beginning and end of the initial editing session and then the editing session for deleting the end of the journal file. The last EDIT/EDT command starts the editing session that will use the edited journal file.

```
☞ EDIT /EDT LETTER1.RNO
    .
    .
    .
*EXIT /SAVE

☞ EDIT /EDT /NOJOURNAL LETTER1.JOU
    .
    .
    .
*DELETE REST
*EXIT

☞ EDIT /EDT /RECOVER /JOURNAL=LETTER1.JOU LETTER2.RNO
```

Notice the use of the /NOJOURNAL qualifier in the second EDIT/EDT system command line. This qualifier prevents EDT from adding on to the existing journal file while you are editing it. See the system appendix for your operating system to find out more about the /RECOVER and /[NO]JOURNAL qualifiers.

7.7 EDT's Tabbing Facility

This section first describes the various functions and commands used in the screen modes to perform tabbing. The line mode TAB ADJUST command is explained afterwards. See Chapter 4 for information on using the TAB key in line mode.

EDT's tabbing facility has two distinct capabilities. You can use EDT's preset tab stops to arrange text in columns. The keypad TAB function and the nokeypad TAB command move text to these positions. In line mode you press the TAB key to move the text to EDT's preset tab stops. As long as you want to arrange text in columns that are a multiple of eight spaces apart, TAB can do the job.

You use the second tabbing capability to format outlines and other types of layered text such as indented computer programs. In order to activate this facility you must first issue a SET TAB command; otherwise none of the remaining tabbing commands and functions has any effect on your text. This tabbing facility only indents entire lines of text.

7.7.1 Tabbing Operations in the Screen Modes

This section begins with a discussion of the TAB key and how it is used with SET NOTAB, the default, to arrange text in columns. Then each of the other screen mode tabbing functions is described. Finally an example shows how you can use the tabbing functions to format an outline.

In the screen modes each nokeypad tabbing command corresponds directly to a keypad tabbing key or key sequence. For this reason, the discussion of screen mode tabbing is limited to keypad mode. Table 7-6 lists each keypad function with its corresponding nokeypad command and a brief description of the action taken by EDT.

Table 7-6: Keypad and Nokeypad Tabbing Commands

Keypad	Nokeypad	Function
CTRL/I TAB	TAB	Move the character that the cursor is on and the characters to the right of the cursor to the nearest EDT preset tab stop.
CTRL/T	TADJ	Tab ADJust: moves the selected range of lines over to the current tab stop.
CTRL/A	TC	Tab Compute: sets the tab level count.
CTRL/D	TD	Tab Decrement: reduces the tab level count.
CTRL/E	TI	Tab Increment: increases the tab level count.

Since both the TAB key and CTRL/I have exactly the same function in keypad mode, only the TAB key is described.

There are only two keypad tabbing functions that actually move text: TAB and CTRL/T. The remaining three (CTRL/A, CTRL/D, and CTRL/E) determine the effect that the first two have on your text.

All the functions, except for TAB, require that a SET TAB value be established for your editing session. EDT's default is SET NOTAB. As long as NOTAB is in effect, pressing CTRL/A, CTRL/D, CTRL/E, or CTRL/T has absolutely no effect on the text you are editing.

EDT has automatic tabs set for every eight columns, regardless of any tab stops that might be set for your terminal. When you press the TAB key, EDT inserts a horizontal tab character (CTRL/I) or an appropriate number of spaces into your text. The effect of inserting a horizontal tab character or spaces is that the text from the cursor character to the end of the line moves to the nearest preset tab stop: column 9, 17, 25, 33, 41, 49, 57, 65, and so on. The TAB key always moves text to the right, no matter what EDT's current direction is. You can use a delete function to move the text back toward the left margin.

The remaining tab setting functions require that the cursor be at the left edge of the screen. Only the initial tab stop can be reset, and its value must be a multiple of the SET TAB value. After you have pressed TAB to move text to the first tab stop, a subsequent use of the key moves the text to the next preset (multiple of eight) tab stop.

The line mode SET TAB command determines the initial tab position used by all the screen mode tabbing functions. When you establish a SET TAB value, you affect only the initial tab stop on a line. Using the tab compute (CTRL/A), decrement (CTRL/D), and increment (CTRL/E) functions, you have the flexibility you need to format outlines and other types of layered text.

When EDT moves text to the right in performing one of the tabbing operations, it uses a combination of spaces and horizontal tab characters to fill in the void. The number of spaces, horizontal tabs, or combinations of both depends on how far EDT moves the text and on the SET TAB value. Although it is generally unnecessary to know exactly how many spaces or horizontal tabs are added, you should be aware that deleting one character could move your text back to the left from one to eight spaces.

7.7.1.1 Using the TAB Function to Arrange Text in Columns — The TAB key moves text to the right. The character that the cursor is on and any text to the right of it on the same line are indented. The number of spaces that the text moves to the right depends on the current position of the cursor.

The following example uses the TAB key to move text to EDT's preset tab stops. Note the use of the DELETE key to remove the horizontal tab character that the TAB key inserted and thus to return the text to its original position.

```
This is a line of text.
```

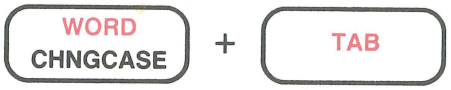
TAB

```
        This is a line of text.
```

DELETE

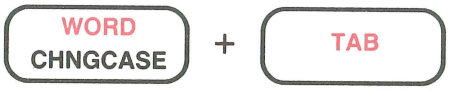
```
This is a line of text.
```

Move the cursor to the **i** in **is** and then press TAB.



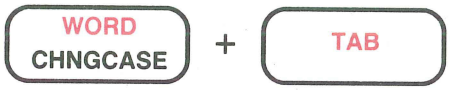

```
This    is a line of text.
```

Move the cursor to the **a** and press TAB.




```
This    is      a line of text.
```

Move the cursor to the **l** in **line** and press TAB.



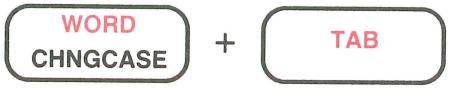

```
This    is      a       line of text.
```

The next example arranges a meeting schedule in columns. You can tab the information as you type it or after you finish. The example shows how to do the tabbing after you have entered the text.

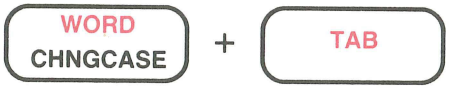
```
Date Meeting Location Time

Jan.13 Relocation Committee Rm.147 2:00 p.m.

Jan.27 Programmer's Seminar Rm.385 1:00 p.m.

Feb.18 Department Staff Rm.143 9:00 a.m.

Feb.27 Division Chiefs Rm.356 2:00 p.m.
```

Start by using the TAB key to move each element over to the nearest tab stop. Adjustments can be made later.

```
Date Meeting Location Time
```




```
Date    Meeting Location Time
```

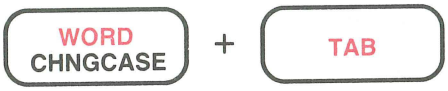



```
Date    Meeting         Location Time
```

```
Date     Meeting          Location         Time
```

Now move the cursor to the **J** in **Jan** and process this line as you did the previous one.

```
Jan.13 Relocation Committee Rm.147 2:00 p.m.
```




```
Jan.13  Relocation Committee Rm.147 2:00 p.m.
```

```
Jan.13   Relocation Committee    Rm.147 2:00 p.m.
```

```
Jan.13   Relocation Committee    Rm.147  2:00 p.m.
```

Use the same functions to reformat the remaining lines. The finished list looks like this:

```
Date     Meeting                 Location          Time
Jan.13   Relocation Committee                Rm.147   2:00 p.m.
Jan.27   Programmer's Seminar                Rm.385   1:00 p.m.
Feb.18   Department Staff                    Rm.143   9:00 a.m.
Feb.27   Division Chiefs                     Rm.356   2:00 p.m.
```

The only adjustment you need to make involves moving the word **Location** over one tab stop. However, since **Location** has eight letters, you must either abbreviate it, change it to another word, or tab the **Time** column over one more stop. In this instance, the easiest solution is to change **Location** to **Room** and tab it over one.

```
Date     Meeting                 Room      Time
Jan.13   Relocation Committee    Rm.147    2:00 p.m.
Jan.27   Programmer's Seminar    Rm.385    1:00 p.m.
Feb.18   Department Staff        Rm.143    9:00 a.m.
Feb.27   Division Chiefs         Rm.356    2:00 p.m.
```

7.7.1.2 Creating Layered Text — In creating layered text you must establish the number of spaces you want to indent one layer of text from another. The line mode SET TAB command establishes the tab value.

```
SET TAB n
```

The SET TAB value is used either alone or in multiples to indent whole lines of text.

The two most important things to remember when a SET TAB value is in effect are:

1. The SET TAB value only affects the first indentation on a line.
2. The cursor must be at the left edge of the screen on the line being moved.

The SET TAB command establishes the initial tab stop for the editing session. The tab compute, tab decrement, and tab increment functions use the SET TAB value to adjust the initial tab stop. Once the necessary adjustments have been made, you use the TAB function to actually indent the text the necessary number of spaces. (Note that if a line of text begins with one or more spaces, use of the tabbing functions might not produce the expected results.)

Generally, when you are creating layered text such as an outline or indented computer program, you want each different layer to be a multiple of a certain number of spaces away from the left margin. For example, in an outline, **A.** level text might be indented 5 spaces, **1.** level 10 spaces, and **a.** level 15 spaces. The SET TAB value represents the number of spaces that you multiply to get the various indentation levels. You might use a SET TAB value of 4 or 5 for outlines, and 8 or 10 for small computer programs.

In other applications, you might want to indent a group of lines 20 spaces or some such value. In that case, you do not need to use a multiplier to create different indentation levels; you can simply set the SET TAB value to the exact number of spaces you want to indent the lines.

For layered text, then, the first step is to decide how many spaces you want for the indentation level. Now you can issue the line mode SET TAB command, for example, SET TAB 4.

As in the example of the meeting schedule, you can enter the text first and then go back to do the indenting work, or you can indent as you go along.

When formatting an outline or computer program, you can use the functions that affect the tab indentation level count: CTRL/A (tab compute), CTRL/D (tab decrement), and CTRL/E (tab increment). You can also use the CTRL/T function to indent blocks of lines.

7.7.1.3 CTRL/E – The Tab Increment Function — After you have issued the SET TAB command, you can move the first line to be indented over to the first indentation level by pressing the TAB key. The first indentation level is equal to the SET TAB value.

When you are ready to move a line to the second indentation level (2 × SET TAB value), you need to use the tab increment function to increase the tab level count to the next higher multiple of the SET TAB value. Use CTRL/E (tab increment) to increase the tab level count to 2.

Each time you need to increase the indentation level, use CTRL/E. Then press TAB to actually move the line of text.

To find out the current tab level count, use the line mode SHOW TAB command. It not only displays the current SET TAB value, but also the current tab level count. To determine exactly

how many spaces your line of text will move, multiply the SET TAB value by the tab level count. In the following example, text moves over the equivalent of 12 spaces when you press the TAB key.

```
*SHOW TAB
tab size 4, tab level 3
```

7.7.1.4 CTRL/D – The Tab Decrement Function — CTRL/D performs the opposite function to CTRL/E. It lowers the tab level count by one. It does not move text back toward the left margin. Rather, it lessens the amount that text is moved to the right. If the current indentation level is 3 and you want the next line of text to be level 1, use CTRL/D twice before pressing the TAB key to actually move the text.

7.7.1.5 CTRL/A – The Tab Compute Function — The tab compute function allows you to start with an indentation level greater than 1, without having to enter repeated tab increments. There are three steps to the process:

1. Issue a SET TAB command.
2. Move the cursor to a character position that is an integer multiple of the SET TAB value (for example, 20 or 25 if the tab size is 5).
3. Press CTRL/A.

Once you have established the position with CTRL/A, press TAB to indent the text to that place.

If the cursor position is not evenly divisible by the SET TAB value when you press CTRL/A, EDT displays the message **Could not align tabs with cursor**.

The SHOW TAB command can verify your indentation level setting. The tab size is still the SET TAB value. CTRL/A affects only the tab level count. When you multiply the two numbers, you get the tab compute position.

After you have set the tab compute position, you can either use CTRL/A again to change the tab level count, or use CTRL/E and CTRL/D to adjust it up or down.

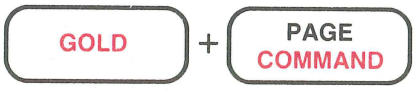
7.7.1.6 CTRL/T – The Tab Adjust Function — The CTRL/T function indents groups of lines using the current SET TAB value. This function is similar to CTRL/A, CTRL/D, and CTRL/E in that it has no effect unless a SET TAB value has been established and it moves only entire lines of text. But CTRL/T differs from these other CTRL tabbing functions in two ways:

1. When you press CTRL/T, the block of text moves to the right without your having to press TAB.
2. The block of text is indented only the amount of the SET TAB value, without regard to the current indentation level. You must use a repeat count to increase the indentation level.

To indent a block of lines using the CTRL/T key sequence, first use the SELECT and LINE functions to create a select range that contains those lines. Then press CTRL/T. (In nokeypad mode, you can use the line or paragraph entity specifier with the appropriate entity count, for example, TADJ4L, or you can establish a select range and use TADJSR.)

EDT indents the range of lines only to the current SET TAB value. To indent text by multiples of the SET TAB value, use the GOLD repeat feature. (For nokeypad, use a repeat count specifier before TADJ, for example, 2TADJSR.)

```
This is the first line.
This is the second line.
This is the third line.
This is the fourth line.
```




```
Command:  SET TAB 7
```

```
This is the first line.
            This is the second line.
            This is the third line.
This is the fourth line.
```

7.7.1.7 Using Screen Mode Tab Functions to Create an Outline – The example in this section creates part of a skeleton outline for a Project Design Seminar. You can start by typing the text. Once that is done, use the various tabbing functions to format it.

```
I.  Identify the Product
A.  Define the Goals
1.  What will the product do?
a.  ----------
b.  ----------
c.  ----------
2.  How will the product perform?
a.  ----------
b.  ----------
c.  ----------
B.  Define the Market
1.  Who will use the product?
a.  ----------
b.  ----------
c.  ----------
2.  How will the product reach the users?
a.  ----------
b.  ----------
c.  ----------
```

You decide to use a SET TAB value of 4 – SET TAB 4.

When you begin to format the outline, you see that the first line stays at the left margin, but the **A.** line needs to be indented one level. TAB takes care of that.

```
I.  Identify the Product
A.  Define the Goals
```

```
I.  Identify the Product
    A.  Define the Goals
```

In order to move the **1.** line over to the second indentation level, use CTRL/E to increment the tab level to 2. Then move the cursor to the first character of the line before pressing TAB.

```
I.  Identify the Product
    A.  Define the Goals
1.  What will the product do?
```

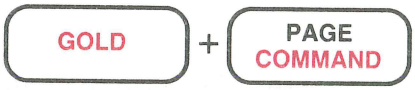
```
I.  Identify the Product
    A.  Define the Goals
        1.  What will the product do?
```

The next step is to move the **a.** level line. Again, use CTRL/E to increase the indentation level.

```
I.  Identify the Product
    A.  Define the Goals
        1.  What will the product do?
a.  ----------
```

```
I.  Identify the Product
    A.  Define the Goals
        1.  What will the product do?
            @.  ----------
```

You can issue the SHOW TAB command to verify the SET TAB value and the tab level count.




```
Command: SHOW TAB
```

```
Command:  SHOW TAB
tab size 4; tab level 3
```

Now move the cursor to the **b.** line and press TAB. Do the same for the **c.** line. Here is the outline so far:

```
I.  Identify the Product
    A.  Define the Goals
        1.  What will the product do?
            a.  ----------
            b.  ----------
            c.  ----------
2.  How will the product perform?
a.  ----------
b.  ----------
c.  ----------
B.  Define the Market
1.  Who will use the product?
a.  ----------
b.  ----------
c.  ----------
2.  How will the product reach the users?
a.  ----------
b.  ----------
c.  ----------
```

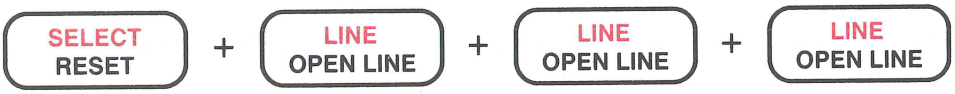
The next line to indent is 2., but that line needs to be moved only 2 levels. Use CTRL/D to change the tab level count from 3 to 2.

```
                c.  ----------
2.  How will the product perform?
```

```
                c.  ----------
2.  How will the product perform?
```

This time, you can move all three third-level lines using CTRL/T with a select range and a repeat count.

```
            2.  How will the product perform?
a.  ----------
b.  ----------
c.  ----------
```



```
            2.  How will the product perform?
                a.  ----------
                b.  ----------
                c.  ----------
```

```
Command: SHOW TAB
```

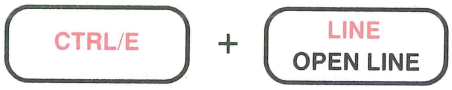
```
Command: SHOW TAB
tab size 4; tab level 2
```

You are still at level 2 because CTRL/T has no effect on the tab level count. But, you need to return to level 1 for the **B.** line.

```
                c.  ----------
B.  Define the Market
```

```
                c.  ----------
B.  Define the Market
```

Now increase the indentation level and move the cursor to the second **1.** line.




```
    B.  Define the Market
1.  Who will use the product?
```

```
    B.  Define the Market
        1.  Who will use the product?
```

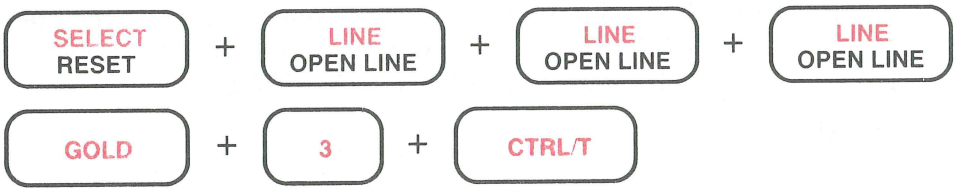
Instead of indenting the next third level group, move the cursor to the second line to indent that line.

```
c.  ----------
2.  How will the product reach the users?
```

```
c.  ----------
        2.  How will the product reach the users?
```

You are now ready to select the last three lines and indent the block with CTRL/T and a repeat count.

```
        2.  How will the product reach the users?
a.  ----------
b.  ----------
c.  ----------
```




```
        2.  How will the product reach the users?
            a.  ----------
            b.  ----------
            c.  ----------
```

You can finish up the indenting by moving the cursor back to the third level block that is not indented, establishing a select range for that block, and pressing GOLD + 3 and then CTRL/T. The final outline looks like this:

```
I.  Identify the Product
    A.  Define the Goals
        1.  What will the product do?
            a.  ----------
            b.  ----------
            c.  ----------
        2.  How will the product perform?
            a.  ----------
            b.  ----------
            c.  ----------
    B.  Define the Market
        1.  Who will use the product?
            a.  ----------
            b.  ----------
            c.  ----------
        2.  How will the product reach the users?
            a.  ----------
            b.  ----------
            c.  ----------
```

7.7.2 Tabbing in Line Mode

The TAB ADJUST command operates EDT's tabbing facility in line mode. The same restrictions that apply to screen mode tabbing also apply to line mode tabbing. You must establish a SET TAB value in order for the TAB ADJUST command to operate. Only whole lines of text can be indented. TAB ADJUST does not arrange text in columns.

The SET TAB command establishes the SET TAB value. EDT's default is SET NOTAB. If you want to check the current SET TAB value before you issue a TAB ADJUST command, type SHOW TAB. EDT displays the current SET TAB value as well as the current tab level.

```
*SET TAB ?
  .
  .
  .
*SHOW TAB
tab size 7; tab level 1
```

The tab level count displayed by the SHOW TAB command refers only to keypad and nokeypad tab levels. The level specifier in the TAB ADJUST command has no effect on tabbing in the screen modes and does not affect the tab level count displayed by SHOW TAB.

The syntax for TAB ADJUST is:

```
TAB ADJUST [-]n [=buffer] [range]
```

The **n** specifier is the tab level count. Notice that **n** is *not* optional. Even if you need a tab level of only one, you must include the **1** in your command statement. The tab level tells EDT how many multiples of the SET TAB value you want the text indented. For example, if the SET TAB value is 7 and the **n** specifier is 3, the text will be indented 21 columns.

If you have text that has already been indented and you want to move it back toward the left margin, precede the level number with a minus sign. If your current text is indented 15 and the current tab size is 5, using -2 brings the text back to the left so that the indentation is only 5 columns.

The buffer specifier enables you to move to another buffer with the TAB ADJUST command. If only a buffer name is specified, EDT indents every line in the buffer. When neither buffer nor range specifier is used, the command operates on the entire current buffer.

Use the range specifier to indicate which lines you want indented. If you want to use TAB ADJUST on several individual lines, you can use commas to separate the different line references. EDT allows up to 19 separate range references per command.

The next example indents the names of the members of a high school social studies department. The original list has no indented lines.

```
12      Social Studies Department
13      Viola R. Burns
14      Lucille Edget
15      Helen J. Weber
16      Howard J. Wise

*SET TAB 10
*TAB ADJUST 1 13 THRU 16

12      Social Studies Department
13                Viola R. Burns
14                Lucille Edget
15                Helen J. Weber
16                Howard J. Wise
```

When you use TAB ADJUST to indent an outline, you can use the string range specifier preceded by ALL to indent several lines at a time, without having to know the line numbers. This technique works best if each outline element is contained on a single line. (Note the SET NONUMBERS command. With SET NUMBERS in effect, when you use the TYPE command to display the outline, some tabs do not appear to have been implemented. SET NONUMBERS allows you to see exactly what EDT has done.)

```
I.  Introduction
A.  Topic
1.  Subtopic
2.  Subtopic
B.  Topic
1.  Subtopic
2.  Subtopic
3.  Subtopic
II. Main Topic
A.  Topic
1.  Subtopic
2.  Subtopic
3.  Subtopic
B.  Topic
1.  Subtopic
2.  Subtopic
C.  Topic
```

```
*SET NONUMBERS
*SET TAB 4
*TAB ADJUST 1 ALL "A."
*TAB ADJUST 1 ALL "B."
*TAB ADJUST 1 ALL "C."

*TAB ADJUST 2 ALL "1."
*TAB ADJUST 2 ALL "2."
*TAB ADJUST 2 ALL "3."
*TYPE WHOLE
```

```
I.  Introduction
    A.  Topic
        1.  Subtopic
        2.  Subtopic
    B.  Topic
        1.  Subtopic
        2.  Subtopic
        3.  Subtopic
II. Main Topic
    A.  Topic
        1.  Subtopic
        2.  Subtopic
        3.  Subtopic
    B.  Topic
        1.  Subtopic
        2.  Subtopic
    C.  Topic
```

7.8 Using Hardcopy Change Mode

This section shows you how to access hardcopy change mode and how to perform some basic editing tasks.

Hardcopy change mode is an editing mode that enables you to use most nokeypad commands on a hardcopy terminal or a screen terminal that is not supported for EDT screen editing. You can use hardcopy change mode if your terminal is set to **Hardcopy**. Use the SHOW TERMINAL command to determine how EDT is interpreting your terminal. Hardcopy change mode is not a separate editing mode in the same sense that keypad, nokeypad, and line modes are distinct. Rather it is a subset of nokeypad mode. Like nokeypad mode, hardcopy change mode is a character editing mode, not a line editing mode.

Most nokeypad commands work in hardcopy change mode. Those that are applicable only to screen terminals, such as REF (refresh), SHL (shift left), SHR (shift right), and TOP, have no effect in hardcopy change mode. The nokeypad HELP and DEFK commands are used only to create keypad key definitions, so you will have no occasion to need them in hardcopy change mode.

The most frequently used nokeypad command that does *not* work in hardcopy change mode is the multiline form of the I (insert) command. You must use the line mode INSERT command to add text to a buffer.

7.8.1 Editing in Hardcopy Change Mode

EDT uses the line mode CHANGE command to shift from line mode to hardcopy change mode. The prompt for hardcopy change mode is C*. Whenever you see that prompt, you can enter nokeypad commands. Line mode commands can only be used if they are preceded by the nokeypad EXT (extend) command. Both CTRL/Z and the nokeypad EX (exit to line mode) command shift EDT from hardcopy change mode to line mode.

EDT uses square brackets ([ ]) to indicate the cursor position in hardcopy change mode. For example, the cursor is at the beginning of the second word on the following line:

```
Accuracy [i]s critical.
```

When the cursor is positioned on the line terminator, EDT shows this by surrounding a <CR> symbol at the end of the line with square brackets.

```
Accuracy is critical.[<CR>]
```

The line terminator symbol appears only at the end of a line when EDT needs to show that the cursor is located at that position.

You can use the single-line version of the nokeypad I (insert) command to insert text. But, if you want to insert several lines at a time, shift to line mode and use the line mode INSERT command. Because CTRL/Z shifts EDT from hardcopy change mode to line mode, you cannot use that key sequence to end your single-line I command. You must type the circumflex (^) character followed by Z.

Here is what happens when you try using CTRL/Z to complete the I command:

```
        Today is [t]he 13th.
C*IFriday^(CTRL/Z)
        Today is Friday the 13th.
*
```

Notice that you are back in line mode. There is no C preceding the asterisk prompt. To get back to hardcopy change mode, you must issue the CHANGE command again. Now try the circumflex function.

```
     Today is [t]he 13th.

C*IFriday▵^Z(RETURN)

     Today is Friday [t]he 13th.
C*
```

7.8.2 A Sample Editing Session in Hardcopy Change Mode

The following sample session takes some text that was entered during a previous EDT session and edits it using hardcopy change mode. Corrections are penciled in on the original version. The corrected version of the text is displayed after you see the hardcopy change mode editing session.

```
June 11, 1984

Smoothe Software Company
Games Division
1234 Bite Boulevard
Houston, TX  77002


Dear Sirs:

I'm writing in response to your ad in the paper for a computer
pgrogrammer.  I feel I have everything you are looking for.

First of all, I'm an acrade junkiue.  I've been addicted to
arcade games since the age of 12.  Ive played TV video games
almost every day since then.  For me, video gams are where the
action is.

Second, I've taken some computer courses so I could make up
my own games on my computer.  I've invented a number of video
games, but I don't have the resources to market the.  If I
could hook up with your company, I could put my games in every
home and every arcade.

Here are some of the games I've written:

PYROMAN      -  See how many homes you can torch and hom
                many fire trucks you can blow up.

ASSASSIN     -  See how many top world leaders you can
                bump off in a month.  blow away in the allotted time.

SURVIVAL     -  See how many World War II survivors you can
                keep from invading your nuclear bomb shelter
                for food and medication.
                                your 5-year supply of
BEETLES      -  See how many gaint maneating beetles you can
                bump off before they eat you up.

Let me know when I can com for my interview.
                          and where
Yours in output space,
     from


JohnQ.Smith
```

In this sample session, you make many of the changes with the nokeypad S (substitute) command. Unlike line mode's SUBSTITUTE command, S causes EDT to find the next occurrence of the search string in the current EDT direction. The sample session display has blank lines in it to make it easier to read.

```
☞ EDIT /EDT SMOSOFT.LET

     1          June 11, 1984
*CHANGE
[J]une 11, 1984

C*S/bite/Byte/
1234 Byte[ ]Boulevard

C*'in the paper'
I'm writing in response to your ad [i]n the paper for a computer

C*D3W
I'm writing in response to your ad [f]or a computer

C*'gpr'
String was not found
I'm writing in response to your ad [f]or a computer

C*'pgr'
[p]grogrammer.  I feel I have everything youa re lloking for.

C*CDC
p[r]ogrammer.  I feel I have everything youa re lloking for.

C*6W DEL
programmer.  I feel I have everything [<CR>]

C*Iyou are looking for.^Z
programmer.  I feel I have everything you are looking for.[<CR>]

C*S/acr/arc/
First of all, I'm an arc[a]de junkiue.  I've been addicted to

C*S/iue/ie/
First of all, I'm an arcade junkie[.]  I've been addicted to

C*S/ive/I've/
arcade games since the age of 12.  I've[ ]played TV video games

C*S/gams/games/
almost every day since then.  For me, video games[ ]are where the

C*S/ket the/ket them/
games, but I don't have the resources to market them[.]  If I

C*S/dould/could/
could[ ]hook up with your company, I could put my games in every
```

```
C*6EL
PYROMAN     -  See how many homes you can torch and hom[<CR>]

C*-DC Iw^Z
PYROMAN     -  See how many homes you can torch and how[<CR>]

C*'bump'
                [b]ump off in a month.

C*DEL
                [<CR>]

C*Iblow away in the allotted time.^Z
                blow away in the allotted time.

C*S/II/III/
SURVIVAL    -  See how many World War III[ ]survivors you can

C*'foo'
                for [f]ood and medication.

C*Iyour 5-year supply of ^Z
                for your 5-year supply of [f]ood and medication.

C*S/gaint/giant/
BEETLES     -  See how many giant[ ]maneating beetles you can{

C*4C I-^Z
BEETLES     -  See how many giant man-[e]ating beetles you can{

C*EL -DC
BEETLES     -  See how many giant man-eating beetles you can[<CR>]

C*' for my'
Let me know when I can com[ ]for my interview.

C*Ie^Z
Let me know when I can come[ ]for my interview.

C*-3w Iand where ^Z
Let me know when and where [I] can come for my interview.

C*S/your in/Yours from/
Yours from[ ]output space,

C*'q.' D2C I Q.^Z
John Q. [S]mith

C*EXT EXIT
DISK$USER:[SMITH.LETTERS]SMOSOFT.LET;2 46 lines
```

Here is the corrected version:

```
June 11, 1984

Smooth Software Company
Games Division
1234 Byte Boulevard
Houston, TX  77002


Dear Sirs:

I'm writing in response to your ad for a computer
programmer.  I feel I have everything you are looking for.

First of all, I'm an arcade junkie.  I've been addicted to
arcade games since the age of 12.  I've played TV video games
almost every day since then.  For me, video games are where the
action is.

Second, I've taken some computer courses so I could make up
my own games on my computer.  I've invented a number of video
games, but I don't have the resources to market them.  If I
could hook up with your company, I could put my games in every
home and every arcade.

Here are some of the games I've written:

PYROMAN     -  See how many homes you can torch and how
               many fire trucks you can blow up.

ASSASSIN    -  See how many top world leaders you can
               blow away in the allotted time.

SURVIVAL    -  See how many World War III survivors you can
               keep from invading your nuclear bomb shelter
               for your 5-year supply of food and medication.

BEETLES     -  See how many giant man-eating beetles you can
               bump off before they eat you up.

Let me know when and where to come for my interview.

Yours from output space,


John Q. Smith
```

7.9 Using EDT Line Numbers

EDT uses line numbers to identify lines in the text you are editing. EDT assigns a number to each line of text regardless of which editing mode you are using or whether SET NONUMBERS is in effect. However, line numbers are only visible in line mode and only when SET NUMBERS (the default) is in effect. (Line numbers are also displayed when you issue certain line mode commands directly from a screen mode.)

EDT issues an identifying number to each line in each buffer. The numbers are maintained in ascending order and each number within a buffer is unique. Although it is possible to force EDT to assign duplicate line numbers with the RESEQUENCE command, this is usually not done and never by EDT itself. Generally, when you add text to a buffer, either by inserting or using a line mode command, such as COPY, INCLUDE, or MOVE, EDT assigns line numbers with decimal fractions to the new lines so that all line numbers are still unique and in ascending order. If more lines are added than there are decimal places available, EDT renumbers subsequent lines to preserve the uniqueness and ascending order.

The EDT line numbers are not part of the text you are editing. Because the line numbers are not text elements, they cannot be edited. Thus, you cannot use them as labels in a BASIC or FORTRAN computer program. EDT line numbers exist for EDT to keep track of each line in a buffer. The line numbers can help you to specify line mode ranges and to find out how many lines you have in a buffer.

7.9.1 Properties of Line Numbers

There are various maximum and minimum limits associated with the EDT line numbering system. The values for EDT line numbers are:

0.00001	smallest possible line number in an EDT session
2814749767.00000	largest possible line number in an EDT session
2814749767	maximum number of lines that can be read into the MAIN buffer by EDT
28147	maximum number of VFC pages for the sequenced file being edited
65535	maximum line number that can be written to the sequence number field of a VFC sequenced file
65535	maximum value for both the :initial and :increment specifiers used with the /SEQUENCE qualifier
65535	largest number of blocks in the work file (If your available disk space is less, then your editing session is limited to that number of blocks.)

7.9.2 EDT Line Numbering with Existing Files

EDT follows preset procedures with regard to line numbering when it copies a file into the MAIN buffer to start your editing session.

- If the input text is not a sequenced file, EDT assigns line numbers starting with 1 and incrementing by 1 until the maximum (2814749767) is reached or until the end of the file.
- If the input text is a sequenced file with all of its sequence numbers unique and in ascending order, EDT uses those sequence numbers as the EDT line numbers.
- If the input text is a sequenced file, but has duplicate sequence numbers or sequence numbers that are out of order, EDT adds a multiple of 100000 starting at the point where the sequence numbers become "unordered" to ensure that the numbers remain unique and ascending.

When you use the INCLUDE command to copy a sequenced file into your editing session, EDT disregards the sequence numbers and assigns numbers that will maintain the uniqueness and ascending order of all lines in the buffer. Even when the INCLUDE command puts the text in an empty buffer, EDT still ignores any sequence numbers.

If your primary input file has more than 2814749767 lines, EDT copies that number of lines and then closes the input file. If you are using INCLUDE to add text to a buffer that already has some text in it, EDT reads in as many lines as possible and then closes the input file. In both cases, a message notifies you that EDT was unable to copy the entire file. The buffer contains all the lines EDT managed to copy before running out of room.

7.9.3 VFC Record Format

A VFC (variable with fixed-length control) record is a line of data that consists of a fixed control area, whose length is the same for all records, and a variable area that can store from zero to a set maximum amount of data. When you use the /SEQUENCE qualifier with the WRITE and EXIT commands, EDT copies the text to the output file in VFC format, placing a sequence number in the fixed control area of the record. The text is stored in the variable portion. The /SEQUENCE qualifier uses the EDT line numbers to create the sequence numbers for the external file.

Later, when you go to edit that file, EDT converts the sequence numbers to EDT line numbers as it puts a copy of the file into the MAIN buffer.

7.9.4 Using the /SEQUENCE Qualifier

The RESEQUENCE, EXIT, and WRITE commands all take the /SEQUENCE qualifier, which can have two specifiers. The syntax is:

```
/SEQUENCE [:initial [:increment] ]
```

You must precede each specifier with a colon (:). In order to use **:increment**, you must also supply **:initial**.

When the /SEQUENCE qualifier is used with the EXIT or WRITE command, only whole numbers (integers) can be entered for the specifiers. Decimal fractions are permitted with the RESEQUENCE command. However, if you use a value that is less than one for either specifier, you must precede the decimal point with a zero.

```
RESEQUENCE /SEQUENCE :0.1 :0.01
```

The /SEQUENCE qualifier performs different functions when used with the RESEQUENCE command than when used with the EXIT and WRITE commands. For instance, the default values for the **:initial** and **:increment** specifiers are different for RESEQUENCE than for EXIT and WRITE.

The RESEQUENCE command without the /SEQUENCE qualifier produces the same results as RESEQUENCE /SEQUENCE with no **:initial** and **:increment** specifiers. EDT renumbers the lines starting with 1 and incrementing by 1. When you omit /SEQUENCE from the RESEQUENCE command, but specify a range that begins with a line number greater than one, EDT prints this message:

```
Range specified by /SEQUENCE would cause duplicate or non-sequential numbers
```

No resequencing takes place. (See Chapter 4 for more information.)

When the /SEQUENCE qualifier is used with the EXIT or WRITE command, EDT creates the output file in VFC record format; that is, with a sequence number in the fixed control portion of the line record. When EDT copies this file into the MAIN buffer for a subsequent EDT session, these sequence numbers become the EDT line numbers.

If you give no specifiers with EXIT /SEQUENCE, the default for **:initial** is the existing line number for the first line in the MAIN buffer. For WRITE /SEQUENCE, **:initial** is the existing line number of the first line of the buffer or range specified with that command. If the existing initial line number has a decimal fraction, EDT truncates it to whole number format.

If you omit the **:increment** specifier, EDT uses the existing number increments whenever possible. Because only whole numbers can be stored in the fixed record area, EDT usually has to make adjustments. Thus, although the /SEQUENCE qualifier with no specifiers indicates that EDT should use the existing line numbers, most or all of your text might be renumbered in the external file.

Generally, you will want to resequence your text before using the /SEQUENCE qualifier with an EXIT or WRITE command. However, if you use a number with a decimal fraction for either the :initial or :increment specifier with the /SEQUENCE qualifier in your EXIT or WRITE command, EDT truncates the decimals to whole numbers and adjusts the remaining numbers to maintain unique and ascending order.

If the :initial and :increment specifiers are greater than 1, but include a decimal fraction, EDT truncates the numbers to whole numbers. If :initial and :increment are between zero and 1, EDT gives all lines the sequence number 0. These sequence numbers are used by EDT as the EDT line numbers when you next edit that file with EDT. The duplicate line numbers will cause EDT to become confused when you try to edit the text.

7.9.5 Line Numbers with the PRINT Command

The PRINT command uses the EDT line numbers, but in a totally different way from the EXIT or WRITE command. When PRINT creates an external file, it changes the nature of the EDT line numbers. The line numbers appear in the text as text elements, which you can edit in future editing sessions.

The PRINT command does not take the /SEQUENCE qualifier. PRINT uses whatever line numbers EDT has assigned to those lines at the time you issue the PRINT command. If the EDT line number has a decimal fraction, the line number in the external file that PRINT creates has the same decimal fraction. If you want to change the EDT line numbers before issuing the PRINT command, use the RESEQUENCE command first.

The PRINT command can be used to number the names or items in a list. For instance, if you have a waiting list, you might want to number the names so that it is easy to tell people how close they are to the top of the list.

```
 3        John West
 4        Lou Gorman
 8        Judy Castellini
 8.1      Sonya Bell
 9        Dave Goldberg
11        Lauren Cummings
```

```
*RESEQUENCE
*TYPE WHOLE

     1        John West
     2        Lou Gorman
     3        Judy Castellini
     4        Sonya Bell
     5        Dave Goldberg
     6        Lauren Cummings
[EOB]

*PRINT WAITLIST.DAT
```

If you do not resequence the file before issuing the PRINT command, the line numbers will be the same as the EDT numbers for those lines. This is what the file would look like when you used a system command to display it at the terminal:

```
<FF>


     3        John West
     4        Lou Gorman
     8        Judy Castellini
     8.1      Sonya Bell
     9        Dave Goldberg
    11        Lauren Cummings
[EOB]
```

Remember that the PRINT command adds a form feed (<FF>) and two blank lines to your output file for every 60 lines of text. If you use EDT to edit WAITLIST.DAT, this is what the MAIN buffer copy looks like:

```
*TYPE WHOLE
     1    <FF>
     2
     3
     4         1        John West
     5         2        Lou Gorman
     6         3        Judy Castellini
     7         4        Sonya Bell
     8         5        Dave Goldberg
     9         6        Lauren Cummings
[EOB]
```

Notice the numbers in the leftmost column. These are the new EDT line numbers that have been assigned for the new editing session. You can delete or change the numbers in the right-hand column because those numbers are now part of your text.

7.10 Creating or Modifying the HELP File

This section first describes the HELP file structure and format. Then it gives some general information on accessing the existing HELP file, so that you can change it, and on making the new text available for your editing sessions. See the appendix on your operating system to find out how to obtain a copy of the HELP text.

EDT allows you to customize the HELP file to reflect key definitions, EDT macros, or other special features that you want documented. If you are in charge of a group of users who have the same customized version of EDT, you can arrange for them to have access to the same customized HELP text.

There are two basic things you can do with the HELP file:

- Create a totally new HELP file
- Modify the existing version of the HELP text

In order to do either, you need to know about the *structure* of the HELP file as well as the *format*. In creating a new HELP file, you need to be concerned with the structural organization that the operating system expects to use. This information is less important if you are only modifying existing text, but still necessary if you plan major changes. As far as format is concerned, you can create your own format if you are writing a new HELP file. But, you need to understand the existing format if you are making only modifications.

7.10.1 HELP File Structure

The EDT HELP file is arranged hierarchically, like an outline, with subtopics nested within topics. There are three levels in the outline of the existing HELP file. You can add a fourth level if you need to. The text for the HELP command itself must be located at the very beginning of the HELP file.

Each HELP topic or subtopic is assigned a level number. The first level includes HELP itself as well as all the main topics under HELP. In the existing HELP file, the level 1 topics are:

```
CHANGE        CLEAR     COPY      DEFINE    DELETE       EXIT      FILL
FIND          HELP      INCLUDE   INSERT    JOURNAL      KEYPAD    MOVE
PRINT         QUIT      RANGE     REPLACE   RESEQUENCE   SET       SHOW
SUBSTITUTE    TAB       TYPE      WRITE
```

Many of these level 1 topics have level 2 subtopics nested under them. For example, when you ask for HELP DEFINE, you get general information on DEFINE and then are told that additional information is available on KEY and MACRO. Both KEY and MACRO are level 2 subtopics under DEFINE.

The KEYPAD level-1 topic has two level 2 subtopics – VT100, VT52. Each of the level-2 subtopics has a number of level-3 subtopics. Since a level-3 subtopic must be physically located after its level-2 subtopic, the level-3 subtopics that are identical for VT100 and VT52 must be located under both subtopics.

If you do not anticipate using your new HELP file on a VT52 terminal, you can ignore or omit that portion of the HELP file. However, if you want the HELP file to pertain to both types of terminals, remember that there are more editing keys on VT100s than VT52s and a different number of keys on the keypad. Thus, there is no way to make the information identical for both terminals.

If you plan to use your HELP file with both terminal types, you must also remember that VT52 terminals cannot handle the escape sequences that are used in the VT100 HELP file to display reverse video or bold text. Trying to read the VT100 keypad diagram on a VT52 terminal is confusing and might cause system the VT52 to "lockup".

HELP text on nokeypad commands is nested under the CHANGE topic. CHANGE has five level-2 subtopics:

```
ENTITIES    HARDCOPY    KEYPAD    SCREEN    SUBCOMMANDS
```

The text of the KEYPAD subtopic under the CHANGE topic is not the same as the level-1 KEYPAD topic. The subtopic merely gives you some information about how to enter KEYPAD mode using the CHANGE command and tells you how to use the HELP key once you are in keypad mode. Nokeypad command descriptions are level-3 subtopics under the level-2 subtopic SUBCOMMANDS. To get information on the EXT command, for example, you must type:

```
HELP CHANGE SUBCOMMANDS EXT
```

The purpose for the various levels of HELP is so that EDT can list the nested topics or subtopics whenever it displays the HELP text you asked for. By placing the subtopics under their topics, you get this information automatically without having to start over again with a new HELP command. Whenever you add new topics or subtopics to the HELP file, EDT automatically adds their names to the list that is displayed with the next higher level topic.

For example, when you look at the DEFINE text in the HELP file itself, you see:

```
1 DEFINE

The DEFINE command (abbreviated DEF) defines either editing
keys for keypad mode or macros for line mode.
```

When you type HELP DEFINE during an editing session, you see the following:

```
DEFINE

The DEFINE command (abbreviated DEF) defines either editing
keys for keypad mode or macros for line mode.

Additional information available:

KEY         MACRO
```

The last part of the text has been added automatically by EDT because KEY and MACRO are level-2 subtopics located directly after DEFINE in the HELP text.

In the HELP text itself, you must prefix the level number before the topic or subtopic. For example, if you were adding the BELL command to the list of nokeypad commands, you would begin your entry this way:

```
3 BELL
```

You must use a space to separate the level number from the topic or subtopic.

Organization of the topics within the HELP file itself can be done in several ways: alphabetical order, order of most use, or random. For those operating systems that do sequential searches through the HELP text, it is advisable to place the keypad mode help text immediately after

HELP, since that is the most used section of the HELP file. Other operating systems do best with alphabetical order. If time and computer usage are of no concern to you, then random order is acceptable.

Information on which type of organization to use with your operating system is given in the system appendixes.

7.10.2 HELP File Format

When you are creating a new HELP file, you can enter the text any way you like and it will appear that way on the screen. The exception to this is the level number, which must precede each topic or subtopic. The level number never appears on the screen, but must be in the HELP text at the left margin of each topic or subtopic. Remember to use a space to separate the level number from the topic name.

It is not necessary to use blank lines to separate topics or subtopics from one another. The level number is sufficient to signal EDT that the following text is a separate topic.

Try to limit the text for each topic to 20 lines so that the entire topic can be displayed on a terminal screen. Remember, however, that EDT automatically displays the subtopics for each level 1 topic or level 2 subtopic. You might have typed only 20 lines of text, but find the subtopic list spills over to a second screen.

7.10.3 Formatting the Keypad Mode HELP Text

The VT100 keypad mode HELP text takes advantage of the escape sequences available with VT100 terminals to put some of the text in bold or reverse video. You can find information on which escape characters create the various visual effects in your terminal manual. If you are creating a completely new HELP file, you can either use the current version as a sample, create your own visual effects, or leave the text plain as in the VT52 entries.

If you are simply modifying the text, for example, changing the definition for a particular keypad key, you can replace the existing text with new material and leave the escape sequences as they are. In the keypad diagram, which is where escape sequences are most prevalent, you can use EDT to replace the key name with the name you want (for instance **SELECT** can be replaced with **TGSEL**).

The keypad HELP facility is constructed so that users can get HELP information by pressing the appropriate keys on the keypad or main keyboard. In order to achieve this effect in your own HELP file, you must be aware of two things. The first is that pressing the GOLD key after pressing HELP gives you information on GOLD. Therefore, you cannot access information on COMMAND by pressing HELP, then GOLD, then the 7 keypad key. You must include information on both functions associated with the 7 key. When the user presses 7, EDT displays information about both the primary and alternate functions of that key.

The second important item is that you must use the EDT key reference number as the subtopic heading. This enables EDT to reference the text when the user presses the key. If you use the function name as the subtopic, no information will be printed on the screen when the user presses that particular key.

The HELP key reference numbers are based on the key definition numbers, with some minor differences.

keypad key HELP number	= DEFINE KEY number + 300
keyboard key HELP number	= DEC Multinational Character Set decimal value with leading zeros as needed to make a three-place number
LK201 FUNCTION key HELP number	= DEFINE KEY FUNCTION number +400

Thus, if you redefined the SELECT key (DEFINE KEY 16 AS "..."), you would enter the new information in the HELP file under number 316. HELP text for the A key is under number 065.

If you are modifying the existing HELP text, simply be sure to place the new text under the same HELP key number as the text you are replacing. When you add HELP text for additional keys, you must use the HELP key number. Table 7-7 gives some corresponding key numbers.

Table 7-7: DEFINE KEY Numbers / HELP KEY Numbers

Defined Key	HELP Keystroke	DEFINE KEY Name	HELP KEY Number
HELP	HELP	10	310
GOLD	GOLD	20	320
PAGE	7 (keypad)	7	307
GOLD/COMMAND	7 (keypad)	GOLD 7	307
CTRL/A	A	CONTROL A	001
GOLD/A	A	GOLD A	065
GOLD CTRL/A	A	GOLD CONTROL A	001
CTRL/Z	Z	CONTROL Z	026
GOLD/Z	Z	GOLD Z	090
GOLD CTRL/Z	Z	GOLD CONTROL Z	026
CTRL/^	^	CONTROL ^	030
GOLD/[	[	GOLD [	091
GOLD/{	{	GOLD {	123
DELETE	DELETE	DELETE	127
GOLD/DELETE	DELETE	GOLD DELETE	127
F12	F12	FUNCTION 24	424
DO	Do	FUNCTION 29	429

Notice that you press the keyboard key by itself when you want help on the GOLD key. For information on GOLD/A, press the A key on the keyboard. (Remember that digits cannot be defined, nor can the keyboard minus sign (–), so that you can use EDT's GOLD/repeat feature.) You cannot get help information on space or RETURN because the EDT HELP processor uses those for special purposes.

The following sample HELP text (with VT100 escape symbols) might be used if you have defined GOLD/R to be SHR (shift right).

```
3 082
<ESC>[7mSHIFT RIGHT - GOLD/R<ESC>[m

Shifts the screen image one tab stop (8 character positions) to
the right.  This function only works after a SHIFT LEFT has
been performed.  SHIFT LEFT is GOLD/L.
```

7.10.4 Using Different HELP Files in Your EDT Session

The SET HELP command causes EDT to look for the appropriate HELP text in the current or specified directory. The syntax for SET HELP is:

```
SET HELP file-specification
```

Generally the HELP file is located in a system directory, for example:

```
SET HELP [SYSLIB2]EDTHELP.HLB
```

See your system appendix for the location of the default EDT HELP file on your system.

Unless you specify a different HELP file with the SET HELP command, EDT will use the default system HELP file. If you want to use a different HELP file regularly, you can put the appropriate SET HELP command in your EDT startup command file.

```
SET HELP [JONES]EDTJHELP.HLP
```

The file type for HELP files differs for different operating systems. Check the system appendix to find out which file type to use on your system. The compilation method also differs for the different systems that support EDT. Details on some manipulations that might be needed for your HELP file to be accessible by EDT are provided in those appendixes.

You can use the SHOW HELP command any time during your editing session to show which HELP text is currently available. If you find that you want a different HELP file to be used, simply specify that file with the SET HELP command.

```
*SHOW HELP
Help file name:  XXX:[1,2]EDTHELP.HLP;1

*SET HELP SY:[310,1]EDTHELPX.HLP

*SHOW HELP
Help file name:  SY:[310,1]EDTHELPX.HLP
```

Chapter 8
EDT Command Reference

8.1 Introduction

This chapter contains descriptions of the EDT commands, qualifiers, and specifiers. The following two pages contain a cross-reference chart of the keypad functions, line mode commands, and nokeypad mode commands. These are grouped by mode as well as by function. The SET and SHOW commands do not appear in the chart.

Entries for each command, qualifier, and specifier described in the rest of the chapter are arranged in alphabetical order, with no regard to editing mode. Each entry begins on a new page.

At the top outer corner of the page is the entry name and below that the mode(s) from which the entry is used. For line and nokeypad commands, the next item is the syntax statement. Keypad functions show the keypad designations and the keypad keys used for both VT100 and VT52 terminals. The VT100 keys are shown first, in red; the VT52 keys are brown. The additional preset keypad mode function keys that are available on the LK201 keyboard are shown where appropriate. The key symbols on the left indicate the way the function keys are shown in the EDT documentation. The key symbols on the right show the keys as they appear on the various terminal keyboards.

There is a description of the purpose and use of each item. Information on special restrictions, if any, is also included. Related commands, qualifiers, and specifiers are printed in bold to indicate that these items are themselves entries in the reference section.

Where appropriate, one or more examples are provided. Occasionally in keypad mode, you press different keys to access the function on VT100 and VT52 terminals. In these instances the VT100 user input appears in red; VT52 user input appears in brown.

The last element for many entries is Related Commands, which lists commands or keypad functions that are similar to that entry. Related commands are limited to those that perform almost the identical function in a different mode. If no related command is listed, it means that none exists. For example, there are no related commands for any of the SET or SHOW commands.

EDT Cross Reference Chart			
Operation	**Keypad**	**Line**	**Nokeypad**
Changing Case	CHNGCASE		CHGC CHGL CHGU DLWC DMOV DUPC
Command Aborting	CTRL/C	CTRL/C	CTRL/C
Command Editing	DELETE CTRL/U RESET	(DELETE) CTRL/U	(DELETE) CTRL/U
Command Processing	ENTER (Do) GOLD	(RETURN)	(RETURN)
Creating Output Files		EXIT PRINT WRITE	
Defining Keys	CTRL/K	DEFINE KEY	[DEFK] [HELP]
Defining Macros		DEFINE MACRO	
Deleting Text	CTRL/J CTRL/U CUT DEL C DEL EOL DELETE DEL L DEL W LINEFEED	CLEAR DELETE REPLACE	CUT D R
EDT Direction	ADVANCE BACKUP		ADV BACK
Ending EDT Session		EXIT QUIT	QUIT
Formatting Text	CTRL/A CTRL/D CTRL/E CTRL/I CTRL/T FILL TAB	FILL TAB ADJUST	FILL TAB TADJ TC TD TI
Help	HELP	HELP	
Inserting Text	OPEN LINE PASTE RETURN	INCLUDE INSERT REPLACE	I PASTE[KS] R

(Continued on next page)

EDT Cross Reference Chart (Cont.)			
Operation	**Keypad**	**Line**	**Nokeypad**
Inserting Special Characters	SPECINS		ASC DATE
Moving and Copying Text	APPEND CUT PASTE	COPY MOVE	APPEND CUT PASTE
Moving in the EDT Session	BACKSPACE BOTTOM CHAR CTRL/H Down Arrow EOL FIND FNDNXT Left Arrow LINE PAGE Right Arrow SECT TOP Up Arrow WORD	FIND <null> TYPE	Down Arrow Left Arrow "move" Right Arrow Up Arrow
Refreshing Line or Screen	CTRL/R CTRL/W	CTRL/R	REF
Renumbering EDT Lines		RESEQUENCE	
Selecting a Range	RESET SELECT		DESEL SEL SSEL TGSEL CLSS
Shifting Mode	CTRL/Z COMMAND	CHANGE	EX EXT
Shifting Screen Image			SHL SHR TOP
Substituting Strings	SUBS	SUBSTITUTE SUBSTITUTE NEXT	S SN
Undeleting Text	UND C UND L UND W		UNDC UNDL UNDW

ADV (Advance) Command
Nokeypad

Syntax:

```
ADV
```

Description:

The ADV (advance) command sets the direction for subsequent editing work to forward (to the right of the cursor and down toward the end of the buffer).

When ADV is in effect, you can move backward for a single command by preceding that command with a minus sign (–).

ADV can be used with a search string to ensure that the direction of the search is forward. In this case, ADV follows the quoted string ("New Jersey"ADV). The current direction is now forward.

If a keypad key whose definition starts with the ADV command is used to process a prompt, the function of that key is also performed. For example, if you have defined a key to be **ADV BR.**, when you press that key to send the search string to EDT in response to the **Search for:** prompt, EDT not only processes the search, but also the rest of the ADV BR. definition, thus moving the cursor to the top of the buffer.

ADV is the default and remains in effect until you give the **BACK** command.

Example: Sets the direction to forward. Moves the cursor to the period in **Md.** and deletes the period. Uses the minus sign to reverse direction for the CHGCC command.

```
Mr. John H. Jones
Digital Equipment Corporation
6707 Whitestone Road
Baltimore, Md.  21207

ADV
"Md."
2C
DC
-CHGCC

Mr. John H. Jones
Digital Equipment Corporation
6707 Whitestone Road
Baltimore, MD  21207
```

Related Commands: Keypad – ADVANCE

Keypad Designations:

Description:

ADVANCE sets the direction for subsequent editing work to forward (to the right of the cursor and down toward the end of the buffer). **ADV.** is the nokeypad definition for ADVANCE.

ADVANCE is the default direction and remains in effect until you press **BACKUP**.

You can also use the ADVANCE key to set the direction of and process the **FIND** function. Press the GOLD and FIND keys, next enter the string you want to locate, then press ADVANCE to move the cursor forward to find the string.

Example 1: Sets EDT's direction to forward and moves the cursor three words to the right.

```
In regard to your memo of January 15th,
```

ADVANCE BOTTOM + WORD CHNGCASE + WORD CHNGCASE + WORD CHNGCASE

```
In regard to your memo of January 15th,
```

Example 2: Uses ADVANCE to be sure that EDT looks for the string **town** to the right of the cursor.

```
Acton, MA
Bedford, MA
Charlestown, MA
```

GOLD + FNDNXT FIND

```
Search for:  town
```

ADVANCE BOTTOM

```
Acton, MA
Bedford, MA
Charlestown, MA
```

Related Commands: Nokeypad – ADV (advance)

APPEND Function
Keypad

Keypad Designations:

Description:

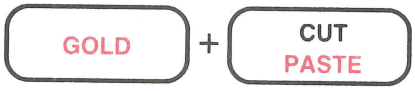
APPEND deletes the select range from the current buffer and adds it to the end of the PASTE buffer. The previous contents of the PASTE buffer are not deleted. **APPENDSR.** is the nokeypad definition for APPEND.

Example: Alphabetizes the list of names using SELECT, CUT, APPEND and PASTE.

```
Joe Spitzer
Trudi Schutz
Becky Santerre
Kathy Waldbauer
```

(Move the cursor to the **T** in **Trudi**.)

(Move the cursor to the **J** in **Joe**.)



SELECT RESET + LINE OPEN LINE + CUT PASTE

(Move the cursor to the **T** in **Trudi**.)

SELECT RESET + LINE OPEN LINE + GOLD + LEFT APPEND

(Move the cursor to the **J** in **Joe**.)

GOLD + CUT PASTE

```
Becky Santerre
Trudi Schutz
Joe Spitzer
Kathy Waldbauer
```

Related Commands: Nokeypad – APPEND

APPEND Command
Nokeypad

Syntax:

```
[+|-][count]APPEND[+|-][count]entity[=buffer]
```

Description:

The APPEND command deletes the specified entity from the current buffer and adds it to the end of either the PASTE buffer (the default) or the specified buffer. The previous contents of the PASTE or specified buffer remain in that buffer; nothing is deleted. The appended entity is inserted below the text that is already in the buffer.

Example: Alphabetizes the list of names using CUT, APPEND, and PASTE.

```
Joe Spitzer
Trudi Schutz
Becky Santerre
Kathy Waldbauer
```

```
CUTL
```

(Move the cursor to the **T** in **Trudi**.)

```
APPENDL
```

(Move the cursor to the **J** in **Joe**.)

```
PASTE
```

```
Becky Santerre
Trudi Schutz
Joe Spitzer
Kathy Waldbauer
```

Related Commands: Keypad – APPEND

Syntax:

```
[number]ASC
```

Description:

The ASC (ASCII) command causes EDT to insert a special character into the text. The inserted character corresponds to the decimal equivalent **number** specified with the ASC command. EDT inserts the character to the left of the cursor. You must use a separate ASC command for each different character you want to insert.

If you do not supply the number specifier with the ASC command, EDT inserts the null character – ^@ (decimal value 00). The number specifier can range from 0 to 255. Appendix G lists characters in the DEC Multinational Character Set with their decimal equivalent values.

In order to use a count specifier with the ASC command, you must enclose the decimal number and the ASC command word in parentheses, for example, 5(12ASC).

The ASC command is most useful when you need to add control and other "nonprinting" characters to your text. You can use the **circumflex** command (^) to insert ASCII control characters (decimal value 0 through 31) in both text and EDT command lines.

Example: Inserts three bell control characters (CTRL/G) at the end of a line.

```
Today is Halloween!
```

```
3(7ASC)
```

```
Today is Halloween!^G^G^G
```

Related Commands: Keypad – SPECINS

BACK (Backup) Command
Nokeypad

Syntax:

```
BACK
```

Description:

The BACK command sets the direction for subsequent editing work to backward (to the left of the cursor and up toward the beginning of the buffer).

After giving the BACK command, you can move forward for a single command by preceding that command with a plus sign (+).

You can use BACK with a search string to set the direction of the search to backward. In this case, BACK follows the quoted string ("New Jersey" BACK). The current direction is now backward.

If a keypad key whose definition starts with the BACK command is used to process a prompt, the function of that key is also performed. For example, if you have defined a key to be **BACK BR.**, when you press that key to send the search string to EDT in response to the **Search for:** prompt, EDT not only processes the search, but also the rest of the BACK BR. definition, thus moving the cursor to the top of the buffer.

To change EDT's direction to forward, use the **ADV** command.

Example: Sets the direction to backward for the "move" command. The plus sign reverses direction for the CHGUW command only.

```
    Mr. Edward Johnson
    Technical Writer
    Digital Equipment Corporation
    Nashua, Nh  03061

BACK
2W
+CHGUW

    Mr. Edward Johnson
    Technical Writer
    Digital Equipment Corporation
    Nashua, NH  03061
```

Related Commands: Keypad – BACKUP

Keypad Designations:

BACKSPACE		BACK SPACE
BACKSPACE		BACK SPACE
BACKSPACE		F12
CTRL/H	(All Terminals)	CTRL + H

Description:

BACKSPACE causes the cursor to move to the beginning of the current line. If the cursor is already at the beginning of a line, pressing BACKSPACE moves it to the beginning of the previous line. **BL.** is the nokeypad definition for BACKSPACE.

The BACKSPACE key and CTRL/H always have the same preset function in EDT. When you redefine the BACKSPACE key, you redefine CTRL/H (except for terminals with LK201 keyboards when they are operating in VT200 mode.) To redefine the BACKSPACE key using the line mode DEFINE KEY command, type **DEFINE KEY** CONTROL H. To find out what the definition of the BACKSPACE key is, type **SHOW KEY** CONTROL H. For terminals with LK201 keyboards, use DEFINE KEY FUNCTION 24 and SHOW KEY FUNCTION 24 for the F12 key.

Example: Moves the cursor to the beginning of the third line and then to the beginning of the second line.

```
All employees wishing to leave early on
December 24th must obtain permission from
their department supervisors.
```

BACKSPACE + BACKSPACE

```
All employees wishing to leave early on
December 24th must obtain permission from
their department supervisors.
```

Related Commands: Nokeypad – BL (beginning of line)

BACKUP Function
Keypad

Keypad Designations:

Description:

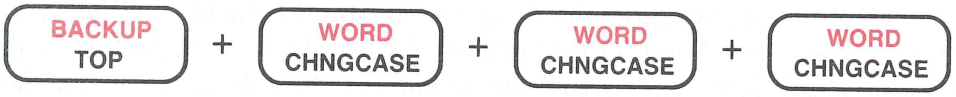
BACKUP sets the direction for subsequent editing work to backward (to the left of the cursor and up toward the beginning of the buffer). **BACK.** is the nokeypad definition for BACKUP.

You can use the BACKUP key to set the direction of and process the **FIND** function. Press the GOLD and FIND keys, enter the string you want to locate, and then press BACKUP to move the cursor backward to find the string.

To change EDT's direction to forward, use **ADVANCE**. The **RESET** function also sets EDT's direction to forward.

Example 1: Sets the direction to backward and moves the cursor three words to the left.

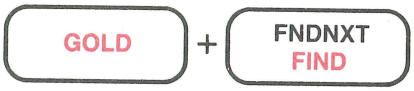
```
In regard to your memo of January 15th,
```




```
In regard to your memo of January 15th,
```

Example 2: Ensures that EDT searches for the string **Ana** toward the top of the buffer.

```
Anaheim, CA
Bakersfield, CA
Claremont, CA
```




```
Search for:  Ana
```

```
Anaheim, CA
Bakersfield, CA
Claremont, CA
```

Related Commands: Nokeypad – BACK (backup)

Syntax:

```
BELL
```

Description:

The BELL command is designed primarily for use in keypad key definitions. It causes the terminal bell (buzzer) to sound when the command is processed.

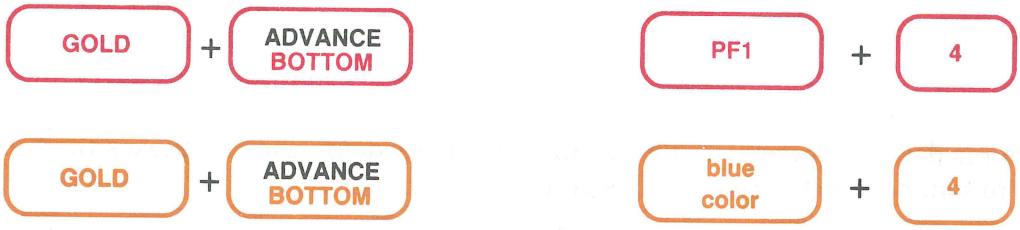
Example: Puts the BELL command in a keypad key definition.

```
*DEFINE KEY GOLD L AS "2500DL BELL."
```

The key sequence GOLD/L is defined to delete 2500 lines. When EDT finishes deleting the lines, it sounds the terminal bell, letting you know that the work is complete.

BOTTOM Function
Keypad

Keypad Designations:



Description:

BOTTOM moves the cursor to the end of the buffer, after the last character position in the buffer. The cursor is now positioned at the end of buffer ([EOB]) mark. **ER.** is the nokeypad definition for BOTTOM.

Example: Moves the cursor to the bottom of the buffer.

```
Dear Sir:
    .
    .
    .
Sincerely yours,
.BLANK 5
David Foster
Manager
[EOB]
```

```
Dear Sir:
    .
    .
    .
Sincerely yours,
.BLANK 5
David Foster
Manager
[EOB]
```

Related Commands: Line – TYPE END
Nokeypad – ER (end of range)

Syntax:

```
/BRIEF[:n]
```

Description:

The /BRIEF qualifier is used with the line mode **SUBSTITUTE** and **TYPE** commands. All EDT qualifiers must be preceded with slashes.

/BRIEF instructs EDT to display only the first **n** characters of a line, instead of the entire line. The default value for **n** is 10.

Example 1: Instructs EDT to display only the first 15 characters of the line in which the substitution took place.

```
    3       Mr. Arlen J. Coolidge
    4       3829 Gardner Avenue
    5       Hollywood, FL  33021

*SUBSTITUTE*J.*R.* /BRIEF:15

    3       Mr. Arlen R. Co
1 substitution
```

Example 2: Instructs EDT to display only the first 10 characters of line 5.

```
    3       Mr. Arlen R. Coolidge
    4       3829 Gardner Avenue
    5       Hollywood, FL  33021

*TYPE 5 /BRIEF

    5       Hollywood,
```

buffer specifier
Line
Nokeypad

Syntax:

```
buffer name
=buffer name
BUFFER buffer name
```

Description:

The buffer specifier consists of two elements, the signal to EDT that the following characters are a buffer name and the buffer name itself. There are three syntax forms. The first syntax form does not have the signal element and can be used only with the **CLEAR** command. You can use the second and third forms – = buffer and BUFFER buffer –in all other line mode commands that can take a buffer specifier. When the buffer specifier is used with nokeypad commands, you must use the = buffer form.

The specifier **buffer** refers to the name of a location used by EDT to store text during an editing session. When you begin your EDT session, a copy of your file is put in the buffer called MAIN. EDT also creates a buffer called PASTE for use with the **CUT**, **PASTE**, and **APPEND** commands. When you exit from EDT using the **EXIT** command, a copy of the MAIN buffer text is put in an external file. All other buffers created during your editing session disappear. If you use **QUIT** to leave EDT, no copy of the MAIN buffer is made.

You create additional buffers during an EDT session each time you specify a new buffer name in a line or nokeypad command. For example, the command **FIND =LIST** creates a buffer called LIST. There are no keypad functions that create new buffers, but you can use line mode commands while still in keypad mode to create new buffers or move from one buffer to another.

The CLEAR command deletes buffers and their contents from your EDT session. In the case of the MAIN and PASTE buffers, only the contents can be deleted, not the buffers themselves. To delete only the contents of buffers other than MAIN or PASTE, use the line mode **DELETE** command.

Buffer names can have over 80 alphanumeric characters. Only letters, digits, and the underscore character (_) can be used to create buffer names. A letter must always be the first character in the name.

You can edit any buffer you create, as well as the MAIN and PASTE buffers. Text can be copied or moved from one buffer to another. The **SHOW BUFFER** command lists all the accessible buffers in your editing session and indicates the current buffer by preceding the name with an equal sign (=).

EDT has several storage areas that you cannot access. These include the delete character, delete line, delete word, search, and substitute buffers. The first three are used only in keypad and nokeypad editing. These buffers contain the most recent character, line, or word deleted by the respective delete functions. The search buffer is used by the various find, substitute, and change case functions. The substitute buffer is used only by substitute commands. Storage areas also exist for entity and prompt definitions. Although you have some control over what goes into these various buffers, you cannot enter them or edit them and their names never appear in the SHOW BUFFER list.

Examples:

```
*FIND =ADDRESS1
```
Moves to the first line of buffer called ADDRESS1. Displays nothing.

```
*COPY 10 THRU 100 TO =TEMP
```
Copies lines 10 through 100 in the current buffer to the buffer named TEMP and moves to TEMP.

```
*INCLUDE DISTLIST.DAT =DISTLIST
```
Copies the external file DISTLIST.DAT into the buffer called DISTLIST and moves to that buffer.

```
*MOVE =HOLD 1 THRU 32 TO 88
```
Deletes lines 1 through 32 in the buffer HOLD and places them above line 88 in the current buffer.

```
*WRITE LETTER.RNO BUFFER EXTRA
```
Copies the entire contents of the buffer EXTRA to the file LETTER.RNO. EDT remains in the current buffer.

```
*CLEAR JUNK
```
Deletes the buffer called JUNK. If JUNK is the current buffer, EDT is transferred to the MAIN buffer.

```
CUT6L=SAVE
```
This nokeypad command moves six lines to the buffer called SAVE. EDT remains in the current buffer.

```
PASTE=PAR10
```
This nokeypad command inserts the contents of the buffer called PAR10 to the left of the cursor.

CHANGE Command
Line

Syntax:

```
CHANGE [=buffer] [range] [;nokeypad-command(s)]
```

Description:

The CHANGE command transfers your editing session from line mode to keypad, nokeypad, or hardcopy change mode. The buffer and range specifiers determine the cursor position when EDT finishes processing the CHANGE command. The default cursor position is at the beginning of the current line in the current buffer.

When the range specifier is a line number, the cursor appears at the beginning of the line. If you use a string for the range specifier, EDT positions the cursor on the character immediately following the string.

You can include nokeypad commands on the CHANGE command line regardless of which change mode you are accessing. A semicolon (;) separates the command word CHANGE and any location specifiers from the nokeypad command(s).

The default mode with CHANGE is keypad for VT100 and VT52 terminals. To go from line to keypad mode, simply issue the CHANGE command. To use nokeypad mode, you must first give the **SET NOKEYPAD** command, then the CHANGE command. Once SET NOKEYPAD is in effect, you must use the **SET KEYPAD** command with the CHANGE command to enter keypad mode. If your terminal's setting for EDT is hardcopy (HCPY), the CHANGE command shifts EDT to hardcopy change mode.

Example 1: Shifts to keypad mode after the EDT session starts off in line mode.

```
☞ EDIT /EDT MEMO.RNO
    1        .FILL.JUSTIFY

*CHANGE

    .FILL.JUSTIFY
    .LEFT MARGIN0
    .RIGHT MARGIN70
    .PAGE SIZE 58,70
         .
         .
         .
```

Example 2: Shifts to nokeypad mode after the EDT session starts in line mode.

```
☞ EDIT /EDT MEMO.RNO
    1       .FILL.JUSTIFY

*SET NOKEYPAD
*CHANGE

    .FILL.JUSTIFY
    .LEFT MARGIN0
    .RIGHT MARGIN70
    .PAGE SIZE 58,70
        .
        .
    .
```

Example 3: Shifts to keypad mode and moves to the buffer TESTING.

```
*CHANGE =TESTING
```

Example 4: Shifts to keypad mode and moves the cursor to the ninth line of the current buffer.

```
*CHANGE 9
```

Example 5: Shifts to keypad mode and moves the cursor to the beginning of the fourth paragraph.

```
*CHANGE ;3PAR
```

Example 6: Shifts to hardcopy change mode. Assumes that the terminal setting is hardcopy (HCPY).

```
*CHANGE
[.]FILL.JUSTIFY
C*
```

Related Commands: Keypad – CTRL/Z
Nokeypad – EX (exit to line mode)

CHAR (Character) Function
Keypad

Keypad Designation:

CHAR SPECINS

3

Description:

CHAR (character) moves the cursor one character in the current direction (forward or backward, depending on whether **ADVANCE** or **BACKUP** is in effect). **C.** is the nokeypad definition for CHAR.

This key is not on the VT52 keypad. However, the left and right arrow keys can be used to move the cursor one character position.

Example: Sets the direction to backward and moves the cursor three characters to the left.

```
This product performs perfectly.
```

BACKUP TOP + CHAR SPECINS + CHAR SPECINS + CHAR SPECINS

```
This product performs perfectly.
```

Related Commands: Nokeypad –C (character)

Syntax:

```
[+|-][count]CHGC[+|-][count]entity
```

Description:

The CHGC (change case) command causes the case of all letters within the specified entity to be changed to the opposite case. For example, if the entity is W (word), **Boston** becomes **bOSTON**.

The **CHGL** (change case lower) command only changes uppercase letters to lowercase. Conversely, the **CHGU** (change case upper) command only changes lowercase letters to uppercase.

When CHGC is used with the SR (select range) entity and there is no active select range, the case of one or more letters will be changed, depending on the cursor location. See the chart under the description of the keypad **CHNGCASE** function for more information.

Example: Changes the case of the **M** in **Maynard** and then changes the case of the entire word so that Maynard has all uppercase letters.

```
Maynard, MA
CHGCC
```

(Move the cursor back to the **m** in **maynard**.)

```
CHGCW
    MAYNARD, MA
```

Related Commands: Keypad –CHNGCASE

CHGL (Change Case Lower) Command
Nokeypad

Syntax:

```
[+|-][count]CHGL[+|-][count]entity
```

Description:

The CHGL (change case lower) command changes all uppercase letters within the specified entity to be lowercase. Letters that are already lowercase remain unchanged.

To change lowercase letters to uppercase, use **CHGU** (change case upper). To change all letters to their opposite case use **CHGC** (change case).

When CHGL is used with the SR (select range) entity and there is no active select range, the case of one or more letters might be changed, depending on the cursor location. See the chart under the description of the keypad **CHNGCASE** function for more information. Remember, however, that for CHGL, only uppercase letters are changed to lowercase; lowercase letters remain the same.

Example: Changes all uppercase letters to lowercase for all words in the line.

```
Word Processing/Small Systems
CHGL3W
word processing/small systems
```

Related Commands: Keypad –CHNGCASE

Syntax:

```
[+|-][count]CHGU[+|-][count]entity
```

Description:

The CHGU (change case upper) command changes all lowercase letters to uppercase within the entity. Letters that are already uppercase remain unchanged.

CHGL (change case lower) changes all uppercase letters to lowercase. **CHGC** (change case) changes all lowercase letters to uppercase and all uppercase letters to lowercase.

When CHGU is used with the SR (select range) entity and there is no active select range, the case of one or more letters will be changed, depending on the cursor location. See the chart under the description of the keypad **CHNGCASE** function for more information. Remember, however, that for CHGU, only lowercase letters are changed to uppercase; uppercase letters remain the same.

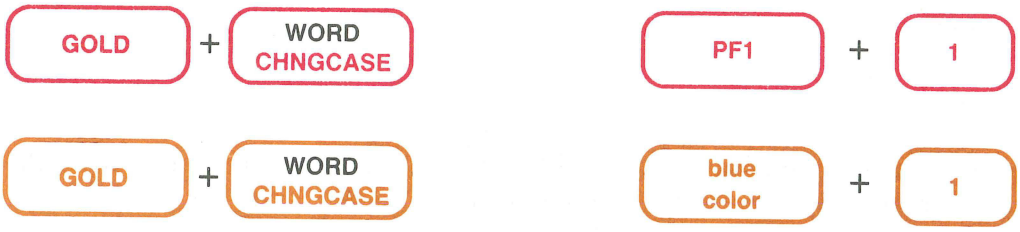
Example: Changes all lowercase letters to uppercase for the entire line.

```
The United States of America
CHGUL
THE UNITED STATES OF AMERICA
```

Related Commands: Keypad–CHNGCASE

CHNGCASE (Change Case) Function
Keypad

Keypad Designations:



Description:

CHNGCASE (change case) changes the case of letters in your text. Uppercase letters become lowercase; lowercase letters become uppercase. The number of letters affected by this function depends on several factors: active select range, cursor location, **SET SEARCH** parameter, and repeat count. The following chart shows what happens when you use CHNGCASE under various conditions:

Conditions	Results
SELECT RANGE ACTIVE	Changes the case of every letter in the select range. All lowercase letters become uppercase; all uppercase letters become lowercase.
NO SELECT RANGE ACTIVE	
1a. SET SEARCH BEGIN in effect. Cursor on first character of current search string. Repeat count = 0 or 1.	Changes the case of every letter in the search string. All lowercase letters become uppercase; all uppercase letters become lowercase. Moves the cursor to the character to the right of the search string.
1b. SET SEARCH END in effect. Cursor to the right of current search string. Repeat count = 0 or 1.	Changes the case of every letter in the search string. All lowercase letters become uppercase; all uppercase letters become lowercase. Moves the cursor to the first character of the search string.
2a. Current direction is forward. *Cursor not at active end of current search string. Repeat count = 0 or 1.	Changes the case of the letter that the cursor is on. Moves the cursor one column to the right.
2b. Current direction is backward. *Cursor not at active end of current search string. Repeat count = 0 or 1.	Changes the case of the letter to the left of the cursor. The cursor remains on the altered letter.

3a. Repeat count greater than 1. Current direction is forward.	Changes the case of the number of letters given in the repeat count. Moves the cursor to the right of the last altered character.
3b. Repeat count greater than 1. Current direction is backward.	Changes the case of the number of letters given in the repeat count. Moves the cursor to the leftmost altered letter.

* Active end of select range:

SET SEARCH BEGIN	Cursor on first character of current search string.
SET SEARCH END	Cursor one character to the right of current search string.

CHGCSR. is the nokeypad definition of CHNGCASE.

Example 1: First changes the case of the **M** in **Maynard** to lowercase. Then changes the case of the entire word to uppercase.

```
Maynard, MA
```

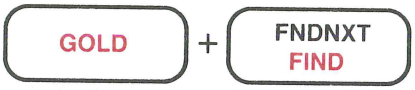
```
m@ynard, MA
```

(Move the cursor back to the **m** in **maynard**.)

```
MAYNARD, MA
```

Example 2: Using FIND and FNDNXT, locates and changes the case of the string **back** in lines 2 and 3.

```
BA[C]K
backspace
backup
```




```
Search for:  back
```

```
BACK
BACKspace
BACK[U]p
```

Related Commands: Nokeypad –CHGC (change case)
CHGL (change case lower)
CHGU (change case upper)

Syntax:

```
[count]^character
```

Description:

The circumflex (^) command enables you to insert ASCII control characters (decimal value 0 through 31) in your text or enter them in nokeypad command lines. When you type the circumflex first and then a character from the keyboard, EDT interprets the two symbols as one control character.

The ASCII control characters are formed from the combination of the control key with the 26 letters and these nonalphanumeric characters: @, [, \,], ^, and _. See Appendix G for the complete list.

You can use the **ASC** command to enter any character from the DEC Multinational Character Set list by using the decimal equivalent values 0 through 255. The circumflex works only for characters with decimal values 0 through 31.

Example: Uses the main keyboard characters circumflex (^) and Z to complete the single line form of the nokeypad **I** (insert) command.

```
Mail to Digital Equipment Corporation↵

I, Merrimack, NH 03054^Z

Mail to Digital Equipment Corporation, Merrimack, NH  03054↵
```

Related Commands: Keypad –SPECINS

CLEAR Command
Line

Syntax:

```
CLEAR buffer
```

Description:

The CLEAR command deletes the entire contents of the specified buffer. The buffer name must be supplied, even if it is the current buffer. When CLEAR deletes the buffer, it removes the buffer name from the list of active buffers for your session. The MAIN and PASTE buffers are exceptions. You can delete the contents of these buffers, but not their names and storage locations. If the buffer you have specified was defined as a macro, the CLEAR command deletes the macro name from the list of valid line mode commands.

When you use CLEAR to delete the current buffer, EDT shifts to the most recent current line in the MAIN buffer.

See the entry on the buffer specifier for more information on how EDT uses buffers.

Example: First deletes the contents and location of the current buffer named DISCARD. Then deletes the contents of the PASTE buffer.

```
*SHOW BUFFER

     =DISCARD          8        lines
      EXTRA   10       lines
      MAIN    256      lines
      PASTE   1        lines

*CLEAR DISCARD

*SHOW BUFFER

      EXTRA   10       lines
     =MAIN    256      lines
      PASTE   1        lines

*CLEAR PASTE

*SHOW BUFFER

      EXTRA   10       lines
     =MAIN    256      lines
      PASTE   No       lines
```

Syntax:

```
CLSS
```

Description:

The CLSS (clear search string) command clears the search string that is currently in the search buffer. After you issue the CLSS command, the search buffer is empty. This command can be used to redefine keypad keys that include SR (select range) in their definitions so that only the select range is affected by the function, never the search string.

When you use the SR (select range) nokeypad entity, EDT first checks to see if there is an active select range. If there is none and the cursor is located in the appropriate position of the current search string, the command that contains the SR entity affects the search string. You might find this inconvenient, particularly with the keypad **CHNGCASE** command, where you often use **FIND** to locate a word in which you only want to change the case of the initial letter. By including CLSS in the CHNGCASE definition, you can empty the search buffer before EDT processes the CHNGCASE function.

Other keypad definitions that include the SR entity are **APPEND**, **CUT**, and **FILL**.

Example 1: Moves to the string **wednesday**. Then clears the search buffer before issuing the CHGCSR command to change the case of the letter **w**.

```
      The meeting will take place on wednesday afternoon at 2:30.

"wednesday" CLSS CHGCSR

      The meeting will take place on Wednesday afternoon at 2:30.
```

Example 2: This example takes place in keypad mode. You first redefine the CHNGCASE function to include the CLSS command ahead of the CHGCSR definition. After finding the string **albu**, EDT changes the case of only the first letter because there is no current search string.

```
      Before leaving for the airport, be sure to contact Jim
      in albuquerque.  If he has not made arrangements for the
```

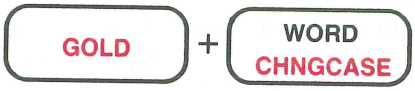
GOLD + PAGE COMMAND

```
Command:  DEFINE KEY GOLD 1 AS "CLSS CHGCSR."
```

ENTER SUBS

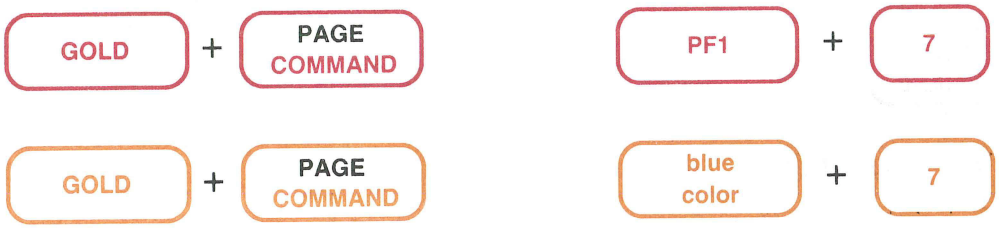
```
Search for:  albu
```

```
     Before leaving for the airport, be sure to contact Jim
     in @lbuquerque.  If he has not made arrangements for the
```




```
     Before leaving for the airport, be sure to contact Jim
     in A1buquerque.  If he has not made arrangements for the
```

Keypad Designations:



Description:

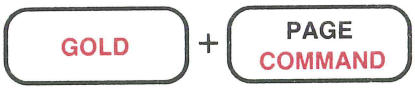
COMMAND is used to enter a line mode command while EDT is still in keypad mode. When you press GOLD and COMMAND, EDT prompts you with **Command:** at the bottom of the screen. Type the line mode command you want to use, for example **EXIT**, then press the **ENTER** key on the keypad to send the command to EDT.

Use **CTRL/Z** to shift from keypad to line editing if you want to issue several line mode commands in a row.

You can issue two or more line mode commands on the same line by separating the commands with semicolons (;). If you want to put nokeypad commands after the line mode command, use the **CHANGE ;nokeypad-command(s)** form. You can use EDT macros with the COMMAND function just as you would any other line mode command.

EXT ?'Command: '. is the nokeypad definition for COMMAND.

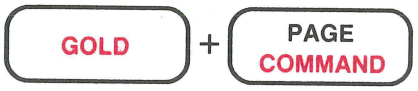
Example 1: Uses the line mode SHOW BUFFER command while still in keypad mode.




```
Command:  SHOW BUFFER
```

```
  BUF1    2         lines
 =BUF2    5         lines
  MAIN    237       lines
  PASTE   1         lines
Press return to continue
```

Example 2: Uses the line mode WRITE command to put a copy of the remaining lines in the current buffer into an external file called SAVE.DAT. EDT remains in keypad mode.




```
Command:  WRITE SAVE.DAT REST
```

```
Command:  WRITE SAVE.DAT REST
DISK$USER:[SMITH]SAVE.DAT 36 lines
```

Related Commands: Nokeypad –EXT (extend)

Syntax:

```
COPY [=buffer-1] [range-1] TO [=buffer-2] [range-2] [/QUERY]
        [/DUPLICATE:n]
```

Description:

The COPY command copies the text specified by location-1 (buffer-1 and/or range-1) and places it directly above the line specified by location-2 (buffer-2 and/or range-2). No text is deleted. You can copy to and from different buffers, creating new buffers as appropriate.

If location-1 is omitted, EDT copies the current line. When location-2 is omitted, EDT inserts a copy of the text above the current line. Whenever you omit the buffer specifier, EDT assumes the current buffer. If buffer-1 is specified without a range-1 specifier, the entire contents of buffer-1 are copied to location-2. If buffer-2 is specified without a range-2 specifier, a copy of the text is inserted at the beginning of buffer-2.

Range-1 can refer to one or more lines in the buffer, but range-2 is limited to a single line in the current or specified buffer. Remember to leave spaces between the buffer name and any range specifier if one is used. Otherwise, EDT might misinterpret the buffer name.

If you do not include a buffer-2 specifier, then EDT completes the COPY command in the same buffer from which you issued the command. When you do specify buffer-2, that buffer becomes the current buffer.

The /**QUERY** qualifier lets you verify each line to be copied to location-2. EDT prompts you with a question mark (?) after displaying a line from location-1 to determine if you want to copy that line. There are four responses to the question mark prompt: Y (YES), N (NO), A (ALL), and Q (QUIT).

The /**DUPLICATE** qualifier enables you to put several copies of the location-1 text above location-2.

Examples:

```
*COPY 11 THRU 14 TO 45
```
Puts a copy of lines 11 through 14 above line 45.

```
*COPY =MAIN TO =OTHER
```
Puts a copy of the contents of the MAIN buffer into the buffer called OTHER. If the OTHER buffer does not already exist, EDT creates it.

```
*COPY 1 TO 8 /DUPLICATE:3
```
Puts three copies of line 1 above line 8.

```
*COPY 4 THRU 7 TO END
```
Puts a copy of lines 4 through 7 in the current buffer after the last line of the current buffer.

```
*COPY "Date" THRU "Dear Sir:"+1 TO =MAIN "Enclosed"
```

Puts a copy of the lines in the current buffer starting with the line containing the word **Date** through the line following the line containing the words **Dear Sir:** immediately above the line containing the word **Enclosed**, located in the MAIN buffer.

```
*COPY 16 THRU 20 TO 58 /QUERY
```

Prints line 16 and then questions you to be sure you want to copy that line. If the answer is Y (YES), line 16 is copied above line 58 and line 17 is displayed. If the answer for line 16 is N (NO), line 16 is not copied; EDT moves to line 17 and displays it. If the answer for line 16 is A (ALL), lines 16 through 20 are copied to the location above line 58 without further prompts. If the answer for line 16 is Q (QUIT), no lines are copied.

```
*COPY =ADDR 12 THRU 16 TO
```

Puts a copy of lines 12 through 16 from the buffer ADDR into the current buffer immediately above the current line.

```
*COPY =MAIN 9 THRU 47 TO =BROWN 9
```

Puts a copy of lines 9 through 47 from the MAIN buffer into the buffer named BROWN immediately above line 9 in that buffer. The new current line is line 9 in the BROWN buffer.

Related Commands: Keypad – CUT + PASTE [+ "move"] + PASTE
Nokeypad – CUT + PASTE [+ "move"] + PASTE

Syntax:

```
digit(s)
```

Description:

The count specifier indicates either the number of times a nokeypad command is to be repeated or the number of entities that the command will affect. The maximum value for the count specifier is 32767.

Examples:

```
5W
```
Moves the cursor five words.

```
CHGC27C
```
Changes the case of the next 27 characters.

```
8UNDC
```
Inserts the contents of the delete character buffer eight times.

```
D4SEN
```
Deletes the next four sentences.

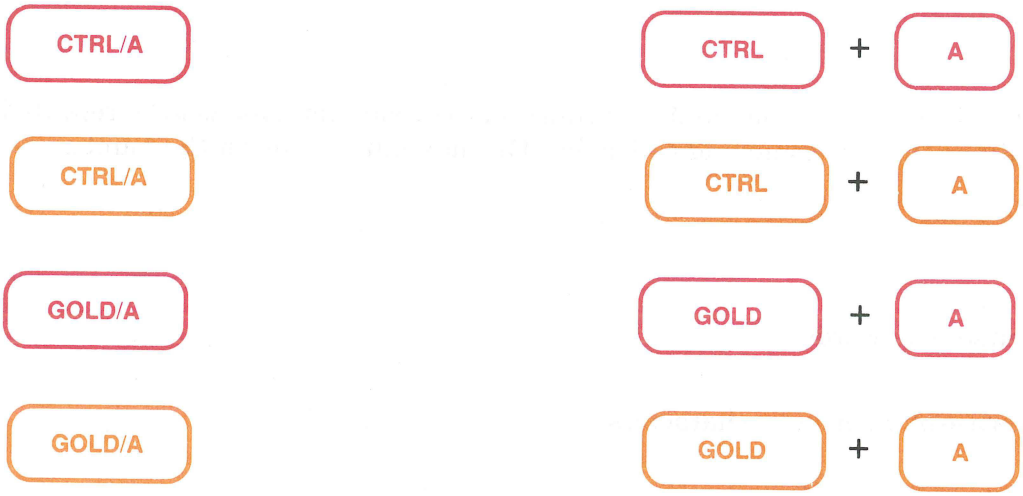
```
-3SN
```
Performs the same substitution three times, but toward the beginning of the buffer.

```
5(+V D+W)
```
Deletes the first word in each of the next five lines. The parentheses indicate that the repeat count applies to both commands – +V and D+W. If you omit the parentheses, EDT deletes only the first word on the fifth line.

```
2(7ASC)
```
Inserts two bell control characters (CTRL/G) into your text. The parentheses are necessary to distinguish the count number from the decimal equivalent value of CTRL/G.

CTRL/A (Control A) Function
GOLD/A
Keypad

Keypad Designations:



Description:

CTRL/A establishes the tab position at the present cursor position and resets the indentation level count to be the quotient of the cursor position divided by the **SET TAB** value. To use CTRL/A, the current cursor position must be a multiple of the SET TAB value. For example, if the SET TAB value is 5, you can use CTRL/A for cursor locations at position 5, 10, 15, 20, and so on. If the cursor is at some other column, for example 13, and you press CTRL/A, EDT prints an error message.

CTRL/A does not move text. You must use the **TAB** function to indent a line. CTRL/A only works if SET TAB is in effect. EDT's default is SET NOTAB. **TC.** is the nokeypad definition for both CTRL/A and GOLD/A.

Example: Using a SET TAB value of 5, sets the indentation level count to 3 to move the first line of text 15 columns to the right. Uses SHOW TAB to verify the tab size and level count.

```
This is the first line.
This is the second line.
```

GOLD + PAGE COMMAND

```
Command:  SET TAB 5
```

ENTER SUBS

(Move the cursor 15 spaces to the right.)

```
This is the first line.
```

CTRL/A

(Now move the cursor back to the beginning of the line you want to indent.)

TAB

```
                This is the first line.
This is the second line.
```

GOLD + PAGE COMMAND

```
Command:  SHOW TAB
```

ENTER SUBS

```
Command:  SHOW TAB
tab size 5; tab level 3
```

Related Commands: Nokeypad – TC (tab compute)

CTRL/C (Control C) Function
Keypad
Line
Nokeypad

Key Designations:

Description:

CTRL/C interrupts certain operations before EDT finishes processing them. You can use CTRL/C to stop a runaway search through a long file or to stop a long repeat count. CTRL/C halts certain EDT operations. For example, you can use CTRL/C to stop EDT from displaying a whole buffer when you have used the line mode **TYPE** command to move to another buffer.

When EDT aborts the operation, it prints the message **Aborted by CTRL/C**. If EDT cannot stop a particular process, it prints the message **CTRL/C ignored**.

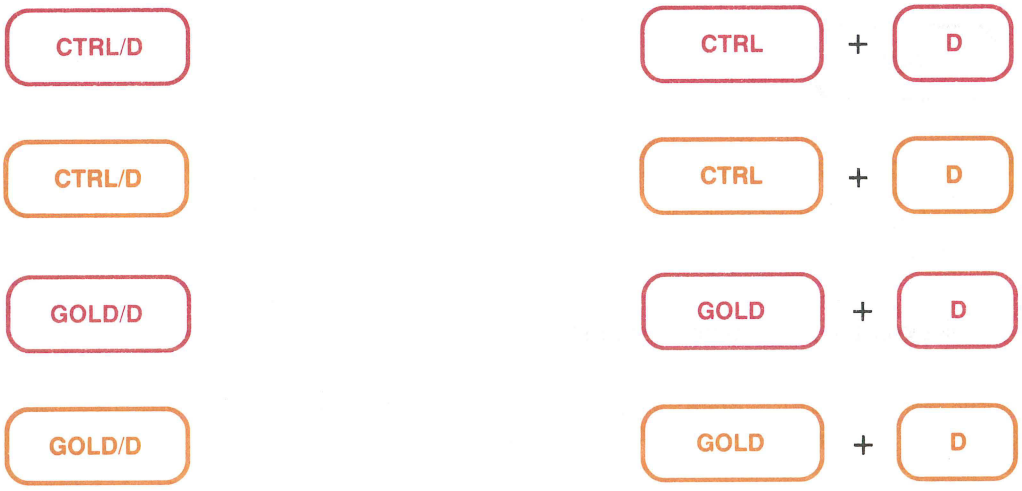
Note: You cannot redefine the CTRL/C function.

Example: Stops EDT from printing the entire contents of the buffer DARCY_LET.

```
*SHOW BUFFER
 DARCY_LET      40      lines
=MAIN   12      lines
 PASTE  No      lines
Press return to continue

*TYPE =DARCY_LET

1       August 20, 1983
2
3       Mr. Charles R. Darcy
4       Production Manager
5       Midland Manufacturing Corporation
6       East St. Louis, IL  62202
```

```
Aborted by CTRL/C
```

Keypad Designations:



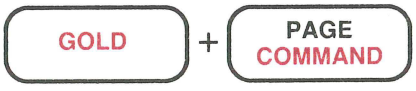
Description:

CTRL/D decreases the TAB level count one tab setting. The tab level count is the multiple of the **SET TAB** value that determines the tab indentation level. Suppose the SET TAB value is 5, the tab level count is 3, and the current indentation level is 15. If you press CTRL/D and then the TAB key, the subsequent text will be moved over 10 columns. The SET TAB value is still 5, but the tab level count is now 2 and the current indentation level is 10.

CTRL/D does not move text. You must use the **TAB** function to indent a line. CTRL/D only works if SET TAB is in effect. EDT's default is SET NOTAB. **TD.** is the nokeypad definition for both CTRL/D and GOLD/D.

Example: First uses SHOW TAB to find out the current tab size and level count. Then decreases the level count by 1 and uses TAB to move the second line four columns to the right.

```
I.  Main Topic
A.  Major Subtopic
1.  Minor Subtopic
2.  Minor Subtopic
```



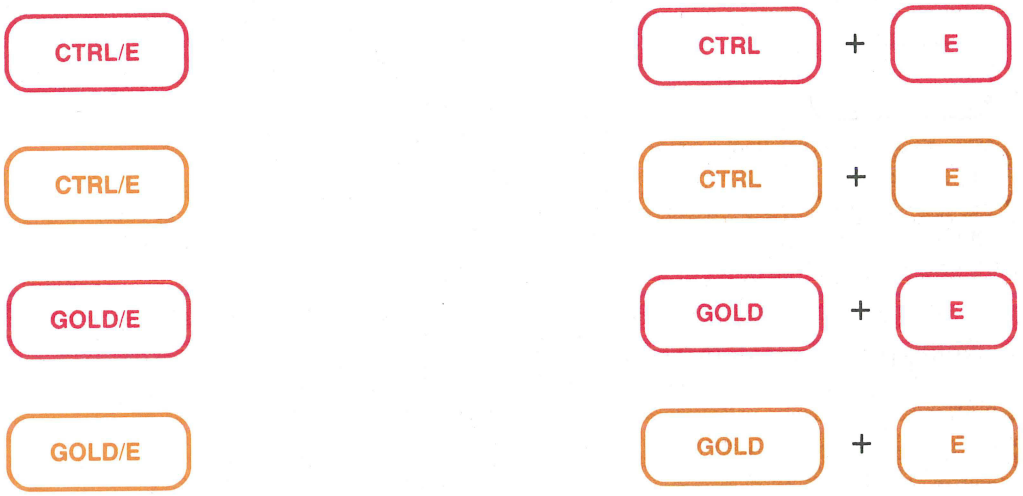

```
Command:  SHOW TAB
```

```
Command:  SHOW TAB
tab size 4; tab level 2
```

```
I.   Main Topic
     A.   Major Subtopic
1.   Minor Subtopic
2.   Minor Subtopic
```

Related Commands: Nokeypad – TD (tab decrement)

Keypad Designations:



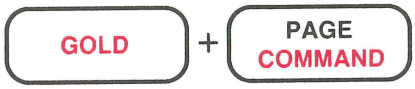
Description:

CTRL/E increments the tab level count by one. The tab level count is the multiple of the **SET TAB** value that determines the tab indentation level. Suppose the SET TAB value is 5, the tab level count is 2 and the current indentation level is 10. If you press CTRL/E and then the TAB key, the subsequent text will be moved over 15 columns. The SET TAB value is still 5, but the tab level count is now 3 and the current indentation level is 15.

CTRL/E does not move text. You must use the **TAB** function to indent a line. CTRL/E only works if SET TAB is in effect. EDT's default is SET NOTAB. **TI.** is the nokeypad definition for both CTRL/E and GOLD/E.

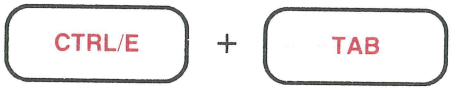
Example: First uses SHOW TAB to find out the current tab size and level count. Then increments the level count by 1 and uses TAB to move the third line eight columns to the right.

```
I.   Main Topic
     A.  Major Subtopic
1.   Minor Subtopic
2.   Minor Subtopic
```




```
Command:  SHOW TAB
```

```
Command:  SHOW TAB
tab size 4; tab level 1
```




```
I.   Main Topic
     A.   Major Subtopic
          1.   Minor Subtopic
2.   Minor Subtopic
```

Related Commands: Nokeypad – TI (tab increment)

Keypad Designations:

Description:

CTRL/K starts the key definition process in keypad mode. You can create key definitions using both nokeypad commands and predefined function keys. You can define a key using only nokeypad commands, by pressing predefined function keys, or by combining nokeypad commands with pressing predefined function keys. **DEFK.** is the nokeypad definition for CTRL/K.

Five types of keys can be defined or redefined:

- **A keypad key with or without GOLD ***
  All keypad keys can be redefined.
- **CONTROL with a keyboard character, with or without GOLD ****
  EDT does not allow you to redefine CTRL/C. Some CONTROL character combinations are system commands and cannot be redefined for that reason. These include O, Q, S, and X. For a complete list of those to avoid on your system, consult the appropriate system appendix.
- **GOLD with a keyboard character**
  GOLD can be used with any keyboard character except the digits 0 through 9 and the minus sign.
- **The DELETE key with or without GOLD*****
  The DELETE key can be redefined by itself or with GOLD.
- **FUNCTION keys on the LK201 keyboard**
  These include the six keys located above the arrow keys on the terminal's "editing" keypad as well as keys F6 through F20 on the function key row across the top of the keyboard.

*If the current definition of a key is **GOLD**, you must use the line mode **DEFINE KEY** command to redefine that key. **GOLD** is the default definition for the top left-hand keypad key (PF1 on VT100; blue on VT52). If you are defining a key to have the RESET function, you must type the word RESET in the definition line.

**You cannot press CTRL/U to enter its definition in the definition line.

***You cannot press DELETE or ⟨X⟩ to enter its definition in the definition line.

To define a key in keypad mode, first press CTRL/K. EDT displays the message:

```
Press the key you wish to define
```

Press the key or key sequence you want to define. EDT then displays the message:

```
Now enter the definition terminated by ENTER
```

Type your definition and/or press existing keypad function keys and then press the ENTER key as instructed. (The DO key on LK201 keyboards does not process a CTRL/K key definition.) The definition process is now complete. If you want to review the definition of any key, use the line mode **SHOW KEY** command.

Any nokeypad command can be used in a key definition. The same entity specifiers that are available in nokeypad mode are available for key definitions. Most preset EDT function keys have nokeypad definitions. (GOLD and RESET are exceptions.) Consult Tables 7-2 through 7-5 or use the SHOW KEY command to see these definitions.

When you look at the definitions for most preset functions, you notice that they end with a period (.). This period is the definition for the **ENTER** function. When you complete your definition with a period, EDT processes the command as soon as you press the key or key sequence. If the period is omitted, EDT stores the command and does not show the results until you press RETURN, or ENTER, or another function key with the period at the end of its definition. For example, if you define GOLD/P to be **2DL.**, when you press the GOLD key and then the P key, two lines will disappear from your screen. If the definition is **2DL**, no change appears on the screen after pressing GOLD and then P. But as soon as you press another function key, the two lines vanish. Be sure to use the period key on the main keyboard, not the one on the keypad, to complete your definitions.

Example 1: Defines the key sequence GOLD/D to insert **Dr.**(SP) to the left of the cursor.

```
James T. Roberts
June C. Rumpole
Andrew R. Schaefer
Cynthia Sears
```

CTRL/K

```
Press the key you wish to define
```

GOLD + D

```
Now enter the definition terminated by ENTER

IDr.▵ CTRL/Z + .
```

ENTER
SUBS

(Move the cursor to the lines you want to modify and press GOLD/D.)

```
    Dr. James T. Roberts
    June C. Rumpole
    Andrew R. Schaefer
    Dr. Cynthia Sears
```

Example 2: Defines the key sequence CTRL/P to remove characters from the end of a line. Using the GOLD/repeat feature, you can process all five lines at once.

```
    **********▯**********
    TW184739-47      $5.76
    TW184741-38      $3.49
    TW184742-84      $7.98
    TW184746-64      $9.85
    TW184747-38      $2.66
```

CTRL/K

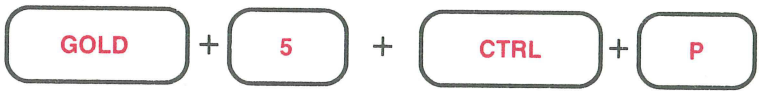
```
Press the key you wish to define
```

CTRL + P

```
Now enter the definition terminated by ENTER

(+V D+EL).
```

ENTER
SUBS

(Position the cursor directly above the space after the 47.)




```
**********************
TW184739-47
TW184741-38
TW184742-84
TW184746-64
TW184747-38
```

Related Commands: Line – DEFINE KEY
[Nokeypad – DEFK]

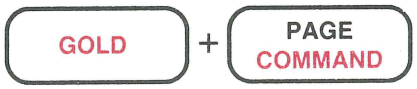
Keypad Designations:

Description:

CTRL/L inserts a form feed character (<FF>) into your text. You can also use CTRL/L to enter a form feed in search strings and SET commands.

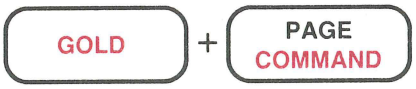
^L. is the nokeypad definition of CTRL/L.

Example 1: Shows how to enter the default boundary list for the SET ENTITY WORD command. The first **Command:** line shows the text you type up to the point where the boundary limits are entered. The second **Command:** line shows what EDT displays on the screen after you have pressed the spacebar and the five control key sequences. The SHOW ENTITY WORD command displays the control characters differently.




```
Command:  SET ENTITY WORD "
```

```
Command:  SET ENTITY WORD " ^I^J^K^L^M"
```




```
Command:  SHOW ENTITY WORD
```

```
Command:  SHOW ENTITY WORD
        <LF><VT><FF><CR>
```

Example 2: Shows how to enter the form feed character in your text.

```
    Page 37
```

```
    <FF>Page 37
```

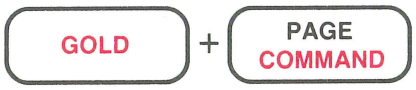
Keypad Designations:

Description:

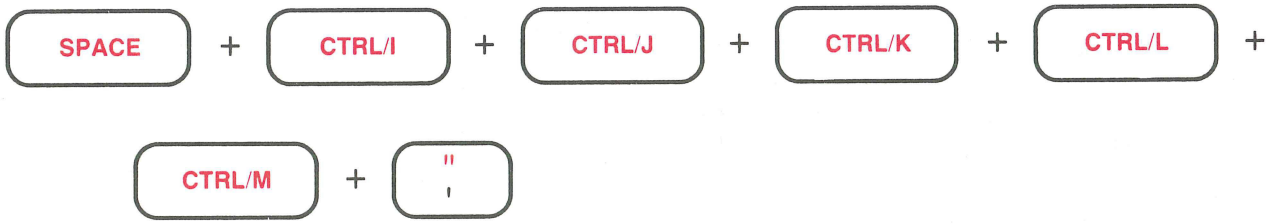
CTRL/M inserts a carriage return character (<CR>) into your text. You can also use CTRL/M to enter a carriage return character (<CR> or ^M) in strings and SET commands. CTRL/M is not identical with an EDT line terminator. However, in keypad mode, you can use CTRL/M to mean a line terminator in search and substitute strings.

^M. is the nokeypad definition for CTRL/M. When you redefine the CTRL/M key sequence, you also automatically redefine the **RETURN** key. It is recommended that you do not alter the preset definition of CTRL/M for that reason.

Example 1: Shows how to enter the default boundary list for the SET ENTITY WORD command. The first **Command:** line shows the text you type up to the point where the boundary limits are entered. The second **Command:** line shows what EDT displays on the screen after you have pressed the spacebar and the five control key sequences. The SHOW ENTITY WORD command displays the control characters differently.




```
Command:  SET ENTITY WORD "
```




```
Command:  SET ENTITY WORD " ^I^J^K^L^M"
```

```
Command:  SHOW ENTITY WORD
```

```
Command:  SHOW ENTITY WORD
        <LF><VT><FF><CR>
```

Example 2: Uses CTRL/M in the search string to locate the string at the end of the line.

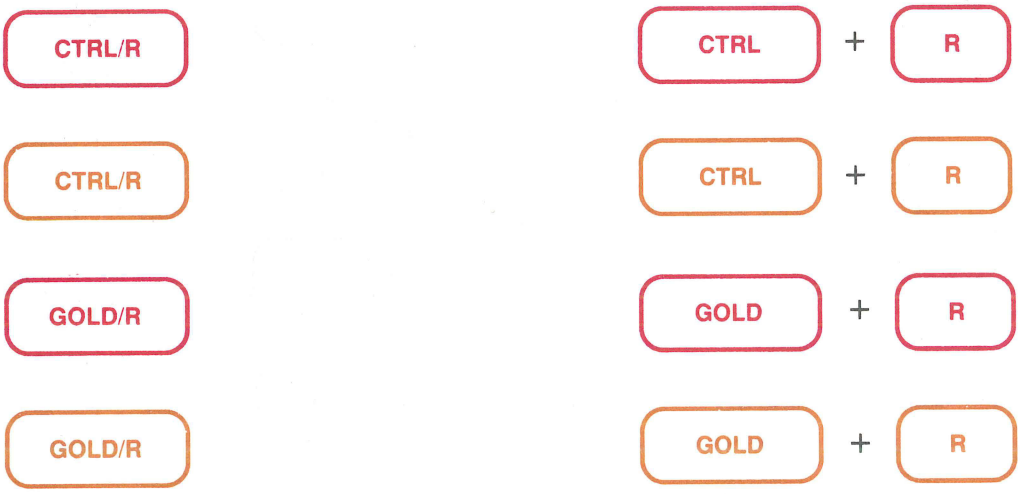
```
    Model Number     Price       Sale Price

    Model 38567X     $38.99      $33.99
    Model 37580Z     $47.99      $41.99
```

```
Search for:
```

```
Search for:  99^M
```

```
    Model Number     Price       Sale Price

    Model 38567X     $38.99      $33.99
    Model 37580Z     $47.99      $41.99
```

Keypad Designations:



Description:

In keypad mode CTRL/R refreshes the screen display. This function has no effect on the text you are editing. It simply clears and redraws the screen, eliminating any extraneous characters or messages that might have appeared on the screen but are not part of the current text you are editing. Note that CTRL/R performs the same function as **CTRL/W** in keypad mode. **REF.** is the nokeypad definition for both CTRL/R and GOLD/R.

In line mode, CTRL/R refreshes the line you are currently typing. If you have used the DELETE key several times to make corrections, you can use CTRL/R to see the characters that are presently part of the line. After using CTRL/R, EDT is still positioned on the line you were typing. You can continue typing on that line or press RETURN to send the data to EDT.

Example: Refreshes the screen to eliminate the notification of new mail on the fourth line.

```
There will be a meeting of the Utilities Team
at 9:30 a.m. on January 15, 1984 in the Glen
Room.  All members are expected to attend.
New mail from XXXXXX::BROWNe served.
```

```
There will be a meeting of the Utilities Team
at 9:30 a.m. on January 15, 1984 in the Glen
Room.  All members are expected to attend.
Coffee and doughnuts will be served.
```

Related Commands: Nokeypad – REF (refresh)

CTRL/T (Control T) Function
GOLD/T
Keypad

Keypad Designations:

CTRL/T

CTRL + T

CTRL/T

CTRL + T

GOLD/T

GOLD + T

GOLD/T

GOLD + T

Description:

CTRL/T indents the lines in a select range which must contain only whole lines. After creating the select range, press CTRL/T to move the select range lines over one tab stop to the right. Use a repeat count to indent the lines more than one tab stop. To move the text one tab stop to the left, press GOLD and then the minus sign (–) before CTRL/T. You can move the lines several tab stops to the left by using both the minus sign and a repeat count.

CTRL/T only works if **SET TAB** is in effect. EDT's default is SET NOTAB. To determine the current SET TAB value, use the **SHOW TAB** command. Note that CTRL/T is not affected by the tab level count, nor does that count have any effect on how far text is indented.

TADJSR. is the nokeypad definition for both CTRL/T and GOLD/T.

Example: Sets the tab value to 5. Using a select range for lines 2 and 3, indents them one tab level (5 columns). Using a select range for lines 4 and 5, and a repeat count of 2, indents them two tab levels (10 columns).

```
I.   Main Topic
A.   Major Subtopic
B.   Major Subtopic
1.   Minor Subtopic
2.   Minor Subtopic
```

GOLD + PAGE COMMAND

```
Command:  SET TAB 5
```

ENTER
SUBS

SELECT
RESET
+
LINE
OPEN LINE
+
LINE
OPEN LINE
+
CTRL/T

SELECT
RESET
+
LINE
OPEN LINE
+
LINE
OPEN LINE

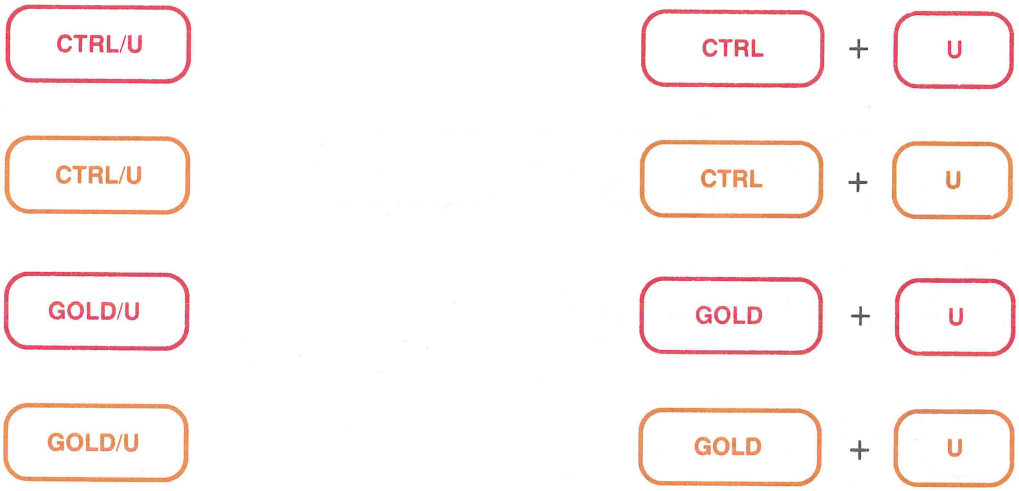
GOLD
+
@
2
+
CTRL/T

```
I.  Main Topic
     A.  Major Subtopic
     B.  Major Subtopic
          1.  Minor Subtopic
          2.  Minor Subtopic
II. Second Main Topic
```

Related Commands: Line – TAB ADJUST
Nokeypad – TADJ (tab adjust)

CTRL/U (Control U) Function
GOLD/U
Keypad

Keypad Designations:



Description:

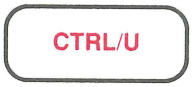
CTRL/U deletes everything from the character to the left of the cursor to the beginning of the line. If the cursor is in the middle or at the end of the line and CTRL/U is pressed, EDT deletes the characters between the cursor and the beginning of that line. If the cursor is at the beginning of a line when CTRL/U is pressed, the line above the cursor is deleted. Text deleted by CTRL/U is stored in the delete line buffer. Use **UND L** to insert or restore the deleted text.

CTRL/U can be used to cancel a **COMMAND**, **FIND**, or **CTRL/K** operation. For example, if you have pressed GOLD/COMMAND and started to type a line mode command, you can press CTRL/U to return the cursor to the text. If you have pressed GOLD/FIND and started to type a search string, you can also press CTRL/U to return the cursor to the text. The string in the search buffer remains the same as it was before you pressed GOLD/FIND. Similarly, if you are in the process of creating a key definition with CTRL/K, you can press CTRL/U to cancel the definition process.

DBL. is the nokeypad definition for both CTRL/U and GOLD/U. CTRL/X always performs the same function as CTRL/U, regardless of the definition assigned to CTRL/X.

Example: Deletes the first half of the second line.

```
Before signing up for the course, check with Fran Pelletier.
Or you can call the registrar yourself.  Be sure that you have
```




```
Before signing up for the course, check with Fran Pelletier.
Be sure that you have
```

Related Commands: Nokeypad – DBL (delete to beginning of line)

Keypad Designations:

CTRL/W	CTRL + W
CTRL/W	CTRL + W
GOLD/W	GOLD + W
GOLD/W	GOLD + W

Description:

CTRL/W or GOLD/W refreshes the screen display. This function has no effect on the text you are editing; it simply clears and redraws the screen, eliminating any extraneous characters or messages that have appeared on the screen but are not part of the current text you are editing. Note that CTRL/W performs the same function as **CTRL/R** in keypad mode. **REF.** is the nokeypad definition for both CTRL/W and GOLD/W.

Example: Refreshes the screen to eliminate the notification of new mail on the fourth line.

```
There will be a meeting of the Utilities Team
at 9:30 a.m. on January 15, 1982 in the Glen
Room.  All members are expected to attend.
New mail from ZZZZZZ::BLACKe served.
```

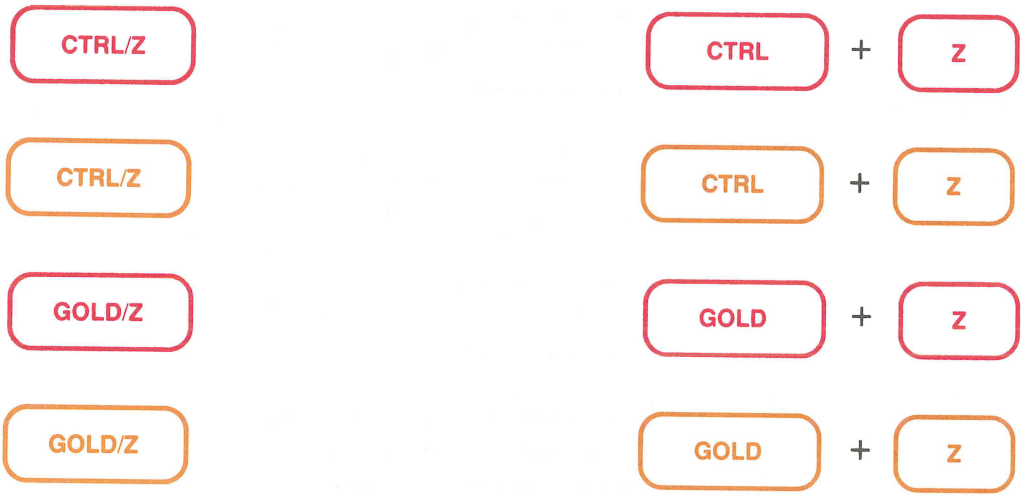
CTRL/W

```
There will be a meeting of the Utilities Team
at 9:30 a.m. on January 15, 1982 in the Glen
Room.  All members are expected to attend.
Coffee and doughnuts will be served.
```

Related Commands: Nokeypad – REF (refresh)

CTRL/Z (Control Z) Function
GOLD/Z
Keypad

Keypad Designations:



Description:

CTRL/Z shifts EDT from keypad mode to line mode. After you have pressed CTRL/Z, the line mode asterisk prompt (*) appears indicating that EDT is ready to accept line mode commands. To resume keypad editing, use the line mode **CHANGE** command.

EX. is the nokeypad definition for CTRL/Z and GOLD/Z.

Example: Shifts from keypad mode to line mode.

```
    Customer called Repair Service 5/25/83.
    One of his telephones was not working.
    He was told to disconnect his phone and
    take it to his nearest phone store.
      .
      .
      .
```

```
*TYPE .
    2           One of his telephones was not working.
```

Related Commands: Line – CHANGE
Nokeypad – EX (exit to line mode)

Key Designations:

CTRL/Z

CTRL + Z

Description:

CTRL/Z performs the same function in line editing as it does in nokeypad editing. (It has a different function in keypad editing.) You use CTRL/Z to exit from the insert state. After you have given the appropriate insert command and typed the text you want to add, use CTRL/Z to complete the insert procedure. CTRL/Z is used with both the **INSERT** and **REPLACE** commands in line mode and the **I** (insert) and **R** (replace) commands in nokeypad mode.

Pressing CTRL/Z three times in succession has the same effect as the line mode QUIT/SAVE command.

Example 1: In line mode, completes the insert operation and returns EDT to the command state. Uses the TYPE command to display the first two lines in the buffer.

```
     1       TO:     Kyle Carlson

*INSERT

             DATE:   March 31, 1986
```

CTRL/Z

```
     ^Z
     1       TO:     Kyle Carlson

*TYPE 0 THRU 1

     0.1     DATE:   March 31, 1986
     1       TO:     Kyle Carlson
```

Example 2: In nokeypad mode, completes the insert operation and returns EDT to the command state.

```
     Customer called Repair Service 5/25/83.
     One of his telephones was not working.
     He was told to disconnect his phone and
     take it to his nearest phone store.

I RETURN
```

(Now type in the line **REPAIR SERVICE REPORT 825** where the cursor is and press RETURN to move the **Customer called** text to the next line.)

```
REPAIR SERVICE REPORT  825
Customer called Repair Service 5/25/83.
One of his telephones was not working.
He was told to disconnect his phone and
take it to his nearest phone store.
```

Keypad Designations:

*On LK201 keyboards, both the 6 keypad key and the **Remove** key have the same preset function.

Description:

CUT removes the active select range from the current buffer and stores it in the PASTE buffer. You can use CUT to delete large or small sections of text. When you use CUT in conjunction with the **PASTE** function, you can move or copy text from one place in the current buffer to another place in that buffer.

When you use CUT to delete only part of a line, EDT adds a line terminator at the end of the text being stored in the PASTE buffer. The line terminator is necessary because EDT cannot store partial lines in the PASTE buffer. When you use the PASTE function, EDT removes the added line terminator. Thus, when you insert the text, you do not have an extra line terminator.

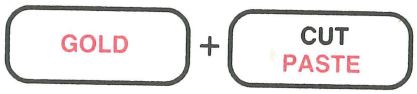
The steps for moving and copying text are described under the keypad PASTE function. **CUTSR.** is the nokeypad definition of CUT.

Example: Using a select range, moves the first line to the PASTE buffer and then uses PASTE to insert that line in the proper chronological sequence between Adams and Madison.

```
Thomas Jefferson
George Washington
John Adams
James Madison
```

(Move the cursor to the **J** in **James Madison**.)

CUT (Cont.)
Keypad




```
George Washington
John Adams
Thomas Jefferson
James Madison
```

Related Commands: Nokeypad – CUT

Syntax:

```
[+|-][count]CUT[+|-][count]entity[=buffer]
```

Description:

The CUT command removes the specified entity from the text buffer and stores it in the PASTE buffer or the specified buffer. EDT uses the PASTE buffer when you do not specify a buffer name with the CUT command.

You can use the CUT command to delete large or small sections of text. When you use CUT in conjunction with the **PASTE** command, you can move or copy text from one place to another in the same buffer, or from one EDT buffer to another, but not to or from external files. The line mode commands **INCLUDE** and **WRITE** are used with external files.

When you use CUT to put part of a line into a buffer, EDT adds a line terminator at the end of the text. The line terminator is necessary because EDT cannot store partial lines in a buffer. When you use the PASTE command to insert that text, EDT removes the extra line terminator.

The steps for moving or copying text are described under the nokeypad PASTE command.

Example: Moves the first line to the TJ buffer and then uses the PASTE command to insert that line in the proper chronological sequence between Adams and Madison.

```
Thomas Jefferson
George Washington
John Adams
James Madison
```

```
CUTL=TJ
```

(Move cursor to the **J** in **James Madison**.)

```
PASTE=TJ
```

```
George Washington
John Adams
Thomas Jefferson
James Madison
```

Related Commands: Keypad – CUT

D (Delete) Command
Nokeypad

Syntax:

```
[+|-][count]D[+|-][count]entity
```

Description:

The D (delete) command removes the specified entity from the current buffer. If the entity involves a character, word, or line, the deleted text is stored in the delete character, delete word, or delete line buffer. EDT overwrites the contents of the appropriate buffer each time you use a D command that involves a character, word, or line entity. You can use the **UNDC**, **UNDW**, and **UNDL** commands to insert the current contents of the respective delete buffers anywhere in the current buffer.

You can use the count specifiers to process several deletions in one command. For example, D3W deletes three words; 2DL deletes two lines. The sign specifier enables you to delete entities to the left of the cursor without changing EDT's direction. For example, –2DC deletes the two characters to the left of the cursor.

Example: Deletes the first line. Then deletes the third line and uses UNDL to insert that line in chronological order. Deletes the last character in **Todd** and the word at the end of the last line.

```
Coming Events:

January 18, 1985      Luncheon Seminar on Technical Communications
January 30, 1985      Meeting of the EDT Development Staff
January 23, 1985      Seminar on Writing for Programmers
February 2, 1985      Lecture by Todd Brockman
February 7, 1985      Meeting of the Secretarial Staff Group
```

```
DL
```

(Move the cursor to the January 23 line.)

```
DL
```

(Move the cursor up to the January 30 line.)

```
UNDL
```

```
January 23, 1985      Seminar on Writing for Programmers
January 30, 1985      Meeting of the EDT Development Staff
```

(Move the cursor to the second **d** in **Todd**.)

```
DC

    February 2, 1985      Lecture by Tod Brockman
```

(Move the cursor to the **G** in **Group**.)

```
DW

    Coming Events:

    January 23, 1985      Seminar on Writing for Programmers
    January 30, 1985      Meeting of the EDT Development Staff
    February 2, 1985      Lecture by Tod Brockman
    February 7, 1985      Meeting of the Secretarial Staff
```

Related Commands: Keypad – DEL C, DEL EOL, DELETE, DEL L, DEL W, LINEFEED, CTRL/U
Line – DELETE

DATE Command
Nokeypad

Syntax:

```
DATE
```

Description:

The DATE command inserts the current date into your text. The format of the DATE text is determined by the operating system. Generally the format includes the time as well as a leading and a trailing space. The text is always placed to the left of the cursor regardless of EDT's current direction.

Example: Inserts the current date and time to the right of the heading.

```
    DATE/TIME:  
DATE

    DATE/TIME:    16-APR-1984 09:33:12 
```

Syntax:

```
DEFINE KEY [GOLD] keypad-key-number AS "string[.]"
DEFINE KEY GOLD character AS "string[.]"
DEFINE KEY [GOLD] CONTROL character AS "string[.]"
DEFINE KEY [GOLD] DELETE AS "string[.]"
DEFINE KEY [GOLD] FUNCTION key-number AS "string[.]"
```

Description:

The DEFINE KEY command defines or redefines function keys used in keypad editing. Key definitions are based on nokeypad commands.

Five types of keys can be defined or redefined:

- **A keypad key with or without GOLD**
  All keypad keys can be defined.
- **CONTROL with a keyboard character, with or without GOLD**
  EDT does not allow you to redefine CTRL/C. Some CONTROL character combinations are system commands and cannot be redefined for that reason. These include O, Q, S, and X. For a complete list of those to avoid on your system, consult the appropriate system appendix.
- **GOLD with a keyboard character**
  GOLD can be used with any keyboard character except the digits 0 through 9 and the minus sign.
- **The DELETE key with or without GOLD**
  The DELETE key can be redefined by itself or with GOLD.
- **FUNCTION keys on the LK201 keyboard**
  These include the six keys located above the arrow keys on the terminal's "editing" keypad as well as keys F6 through F20 on the function key row across the top of the keyboard.

When using the DEFINE KEY command you must type the words GOLD and CONTROL in your commands. Pressing the GOLD or CTRL keys has no effect in line mode. If you are redefining the DELETE key, you must type the word DELETE. To redefine a function key on the LK201 keyboard, type the word FUNCTION.

When you want to change the definition for the **BACKSPACE**, **LINEFEED**, or **TAB** key, you must redefine its CONTROL equivalent: CONTROL H for BACKSPACE, CONTROL J for LINEFEED, and CONTROL I for TAB. For terminals with LK201 keyboards, you can redefine the F12 (BACKSPACE) and F13 (LINEFEED) keys using their function numbers, FUNCTION 24 and FUNCTION 25.

Keypad-key-number refers to EDT's numerical designations for the keypad keys. For example, the period key on the keypad has number 16, the ENTER key has number 21, and the down arrow key has number 13. Figure 7-1 gives the keypad key numbers for each key on both the VT100 and VT52 keypads.

FUNCTION keys are the additional keys on the LK201 keyboard that you can define. These include the six keys located above the arrow keys on the "editing" keypad, as well as the keys on the function key row from F6 through F20. You cannot define keys F1 through F5 on the function key row. EDT uses its own numbers for these keys ranging from FUNCTION 1 through FUNCTION 99 to encompass the special user define keys (UDKs) that are allowed on terminals with LK201 keyboards. Figure 7-3 shows the FUNCTION key numbers for the "editing" keypad keys as well as the function key row.

String is the key definition. Key definitions are composed of nokeypad commands and specifiers. The definition for a single function key can contain several nokeypad commands. An individual nokeypad command can be separated from another nokeypad command with a space, for example **3W 4C 2DC.** or **+V BL I$^Z.**. However, you cannot have spaces within a single nokeypad command. (D3W is valid; D 3W is not.) Consult Tables 7-2 through 7-5 in Chapter 7 or use the **SHOW KEY** command to find out the definitions of EDT's preset function keys.

The optional period determines whether the defined function is automatically processed by simply pressing the key or key sequence. If you omit the period, EDT waits until it receives the **ENTER** signal before processing the function. (The period (.) is the definition of ENTER.) Most preset key definitions end with the period. The exceptions are **GOLD** and **RESET**, which are not nokeypad commands. If you include the period in your definition, be sure to use the keyboard period key, not the keypad one.

If your definition includes more than one nokeypad command, you might want to enclose the definition in parentheses. The parentheses enable you to use the keypad GOLD/repeat feature to process your function more than once. If you try to use a repeat count on a multicommand definition with no parentheses, EDT only repeats the first command. Be sure to put the period outside the parentheses, though. For example, **(BL V W DW).** deletes the second word in a line. You cannot use a repeat count with the nokeypad **EXT** (extend) command.

Examples:

```
*DEFINE KEY ? AS "+PAR."
```

Moves the cursor forward to the beginning of the next paragraph .

```
*DEFINE KEY CONTROL E AS "EXT EXIT."
```

Exits from EDT completely in one step, without having to press several keys and type EXIT.

```
*DEFINE KEY GOLD P AS "CUTSR PASTE."
```

Deletes the select range and pastes it back in the same place, while retaining a copy of the select range in the PASTE buffer for inserting elsewhere during your editing session.

```
*DEFINE KEY CONTROL A AS "CUTSR I^&^Z PASTE I\&^Z."
```

Puts RUNOFF underline flags around a select range.

```
*DEFINE KEY CONTROL V AS "I!!!^Z ^M EXT EXIT."
```

Inserts a mark (!!!) in your text and exits from EDT. The mark makes it easy for you to find the place where you were last working.

```
*DEFINE KEY GOLD 3 AS "?'ENTER NOKEYPAD COMMAND: '."
```

Prompts the user with the message in single quotes. When you type the nokeypad command(s) and press ENTER, EDT processes the command(s).

```
*DEFINE KEY GOLD CONTROL B AS "(+V D-C I|^Z).
```

Instructs EDT to move down one line, delete the character preceding the cursor, and then insert the vertical bar character. This can be used to draw vertical lines in your text.

```
*DEFINE KEY 8 AS "+D"
```

Deletes the entity specified by the subsequent key pressed. For example, if the word key is pressed after 8, the next word is deleted. The absence of the period at the end of the definition means that nothing will happen when the 8 key is pressed by itself. The key pressed *after* the 8 will cause a change in the text to occur.

Related Commands: Keypad – CTRL/K
[Nokeypad – DEFK]

DEFINE MACRO Command
Line

Syntax:

```
DEFINE MACRO macro-name
```

Description:

The DEFINE MACRO command is the part of EDT's macro facility that enables you to create new line mode commands for the duration of your editing session. When you issue the DEFINE MACRO command, EDT adds the macro name to the list of valid line mode commands. The macro name must be the same as the name of the buffer that contains the EDT macro. When you issue the macro name as a line mode command, EDT performs all the line mode commands that constitute the macro.

A macro is a group of one or more line mode commands that EDT performs when you issue the macro name as a line mode command. Creating an EDT macro takes several steps. Start by using the **FIND** command to move to the buffer with the macro name. Then insert the line mode command(s) that constitute that macro. You can issue the DEFINE MACRO command either before or after you put the macro text into its buffer. Once the macro text has been entered into its buffer and you have issued the DEFINE MACRO command, you are ready to use the macro name as a line mode command.

Macro names can be the same as existing EDT line mode commands, but in those cases, EDT redefines the command to be that of the macro for the remainder of the session or until the macro is deleted with the **CLEAR** command. To delete a macro, simply use the macro name as the buffer specifier with CLEAR.

A macro can be saved in an external file and copied into a buffer during your editing session with the **INCLUDE** command. You can put a DEFINE MACRO command in your startup command file, along with the text of the macro and the necessary commands to create the macro.

Example 1: Defines and creates a macro called FORM that activates three SET commands. Then returns to the MAIN buffer.

```
*DEFINE MACRO FORM

*FIND =FORM

*INSERT

            SET TAB 5
            SET ENTITY SENTENCE '.?;'
            SET WRAP 70

CTRL/Z

*FIND =MAIN
```

Example 2: Defines and creates a macro called MNH which inserts the text **Merrimack, NH 03054**. Then returns to the MAIN buffer.

```
*DEFINE MACRO MNH
*FIND =MNH
*INSERT
        INSERT ;Merrimack, NH 03054
CTRL/Z
*FIND =MAIN
```

Example 3: Shows the portion of a startup command file that contains line mode commands to create a macro called MEMO. The MEMO macro inserts the heading for a standard memo. The second FIND command at the end of the example returns EDT to the MAIN buffer so you can begin to type the memo text. Lines beginning with exclamation points are comments in the startup command file. Note that the single-line form of the line mode INSERT command must be used to create the macro text.

```
!
! Creates the macro called MEMO, which contains
!   the heading for a standard memo.
!
FIND =MEMO                          !Moves to the macro buffer.
INSERT; INSERT;                     !Blank line
INSERT; INSERT;                     !Blank line
INSERT; INSERT; DATE:
INSERT; INSERT;                     !Blank line
INSERT; INSERT; TO:
INSERT; INSERT;                     !Blank line
INSERT; INSERT; FROM:
INSERT; INSERT;                     !Blank line
INSERT; INSERT; SUBJECT:
DEFINE MACRO MEMO                   !MEMO is now a line mode command.
FIND =MAIN                          !Returns to the first line of MAIN.
!
!End of macro
```

DEFK (Define Key) Command
Nokeypad

Syntax:

```
DEFK
```

Description:

The DEFK command enables you to define a keypad function key or key sequence other than **CTRL/K** to handle the key definition process. The DEFK command does *not* allow you to create keypad definitions from nokeypad mode. In fact, DEFK has no function at all in nokeypad editing.

Example 1: Using the line mode DEFINE KEY command, defines CTRL/D to have the define key function in keypad mode.

```
*DEFINE KEY CONTROL D AS "DEFK."
```

Example 2: Using the keypad CTRL/K function, defines CTRL/D to have the define key function in keypad mode.

```
Press the key you wish to define
```

```
Enter the definition terminated by ENTER

DEFK.
```

Related Commands: Keypad – CTRL/K
Line – DEFINE KEY

Keypad Designations:

Description:

DEL C (delete character) deletes the character on which the cursor is positioned. The cursor stays in the same position, but the remaining characters on the line shift one position to the left. **D + C.** is the nokeypad definition for DEL C.

The deleted character is stored in the delete character buffer. Only one character at a time can occupy that buffer. Each time you delete a character with DEL C or the **DELETE** function, the contents of the delete character buffer are overwritten. Remember that the delete character buffer is inaccessible to you and that its name does not appear in the list displayed by the **SHOW BUFFER** command.

Use **UND C** to restore or insert the contents of the delete character buffer into your text.

DEL C deletes the character the cursor is on; the DELETE function always deletes the character to the left of the cursor. (**D–C.** is the nokeypad definition for DELETE.)

Example 1: Deletes the second period after **Dr**.

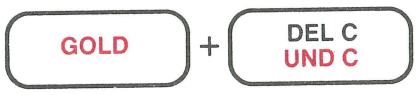
```
Dr.. William W. Williams
```

```
Dr.William W. Williams
```

Example 2: Reverses the order of the mistyped letters, using UND C to insert the deleted letter in its proper place.

```
teh wrong turn
```

(Move the cursor one position to the left.)

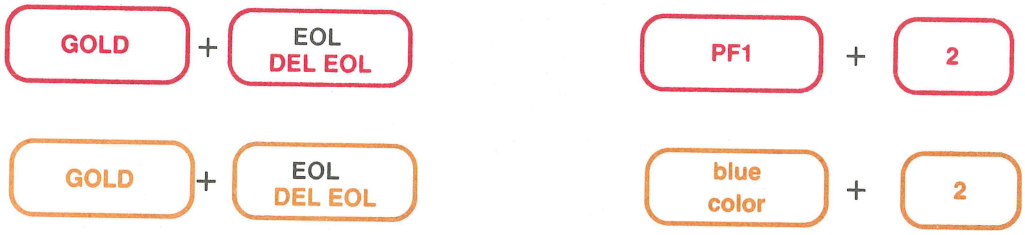



```
the wrong turn
```

Related Commands: Line – DELETE
Nokeypad – DC

DEL EOL (Delete to End of Line) Function
Keypad

Keypad Designations:



Description:

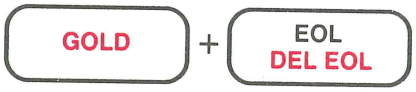
DEL EOL (delete to end of line) deletes everything on a line from the character the cursor is on up to, but not including, the line terminator. The cursor remains in the same position as it was before DEL EOL was pressed. If the cursor is on a line terminator, DEL EOL deletes that line terminator and all the text up to the next line terminator.

The characters deleted from the line are placed in the delete line buffer. Each time DEL EOL, **DEL L**, or **CTRL/U** is used, the contents of that buffer are overwritten. Use **UND L** to restore or insert the contents of the buffer in your text.

When you use DEL EOL, EDT deletes the characters up to the line terminator to the right of the cursor. DEL L deletes those same characters, but also deletes the line terminator and positions the cursor on the first character of the next line. **D+EL.** is the nokeypad definition for DEL EOL. **D+NL.** is the nokeypad definition for DEL L. CTRL/U (**DBL.** is its nokeypad definition) deletes the text from the character to the left of the cursor to the beginning of the line.

Example 1: Deletes the location information after the name.

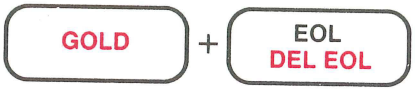
```
Bob Jamison   MK01-2/E37
```




```
Bob Jamison
```

Example 2: Deletes the phone extension number on the second line, but does not delete the line terminator.

```
Al Kerr
x3946
```




```
Al Kerr

```

Related Commands: Line – DELETE
Nokeypad – DEL (delete to end of line)

Keypad Designations:

Description:

The DELETE key deletes the character to the left of the cursor. If the cursor is at the beginning of a line, pressing DELETE deletes the preceding line terminator.

When a character is deleted using the DELETE key, that character is placed in the delete character buffer. The contents of the buffer are overwritten each time a character is deleted either by the DELETE function or by **DEL C**. Use **UND C** to restore or insert the contents of the delete character buffer into the text you are editing.

Use the DELETE key to edit the text you type in response to EDT prompts such as **Search for:** or **Command:**. These deleted characters are not stored in the delete character buffer.

The difference between DELETE and DEL C is that DEL C deletes the character that the cursor is on; DELETE deletes the character to the left of the cursor. **D–C.** is the nokeypad definition for DELETE. **D + C.** is the nokeypad definition of DEL C.

Example 1: Deletes the **c** from the misspelled name.

```
John Smitch
```

```
John Smith
```

Example 2: Deletes the line terminator that is separating the first and second lines.

```
This is the first line.
This is the second line.
This is the third line.
```

```
This is the first line.This is the second line.
This is the third line.
```

Related Commands: Line – DELETE
Nokeypad – D–C (delete character to the left)

Syntax:

```
DELETE [=buffer] [range] [/QUERY]
```

Description:

The DELETE command deletes a line or group of lines, depending on the range that you specify. If no buffer or range is specified, EDT deletes the current line. When you use a buffer specifier but omit the range specifier, EDT moves to that buffer and deletes its entire contents. Whenever you include a buffer specifier, that buffer becomes the current buffer. When you include a range specifier with the DELETE command, EDT deletes all the lines included in that range.

The DELETE command always deletes entire lines, even if the range specifier is a string. In that case, DELETE deletes the line(s) containing that string.

The **/QUERY** qualifier allows you to verify the deletions line by line. When /QUERY is in effect, EDT prints a line and then the question mark prompt (?) to ask if you want that line deleted. The valid responses to the question mark prompt are: Y (YES), N (NO), A (ALL), and Q (QUIT).

Example 1: Deletes the current line.

```
    8       2:30  Meeting with Paul Jeffreys

*DELETE
1 line deleted

    9       3:00 Party for Janice Kay.
```

Example 2: Uses the /QUERY qualifier to delete some, but not all of the lines in the range 1 through 5.

```
    1       Calendar for Tuesday, Dec. 13
    2       Staff Meeting 9:00 a.m., Glen Room
    3       EDT Lecture, 11:00 a.m., Merrimack Auditorium
    4       Luncheon Seminar, 12:00 noon, Hanover Room
    5
    6       Calendar for Wednesday, Dec. 14

*DELETE 1 THRU 5 /QUERY

    1       Calendar for Tuesday, Dec. 13
    ? Y
    2       Staff Meeting 9:00 a.m., Glen Room
    ? Y
    3       EDT Lecture, 11:00 a.m., Merrimack Auditorium
    ? N
    4       Luncheon Seminar, 12:00 noon, Hanover Room
    ? A
4 lines deleted

    6       Calendar for Wednesday, Dec. 14
```

Example 3: Moves to the buffer ADDRS and deletes lines 1 through 4 in that buffer.

```
DELETE =ADDRS 1 THRU 4

    5       Dr. Janet Townley
```

Related Commands: Keypad – DEL L
Nokeypad – DL (delete line)

Keypad Designations:

Description:

DEL L (delete line) deletes everything on a line starting with the character the cursor is on up to and including the line terminator. The cursor position remains unchanged on the screen.

If the cursor is on the first character of the line, the entire line is deleted. The cursor is now positioned on the first character of the following line.

The characters deleted by DEL L, **DEL EOL**, or **CTRL/U** are stored in the delete line buffer. Each time a line or piece of line is deleted, the contents of the delete line buffer are overwritten. Use **UND L** to restore or insert the contents of the delete line buffer into the text you are editing.

DEL L always deletes the line terminator to the right of the cursor; DEL EOL only deletes the characters up to that line terminator. **D + NL.** is the nokeypad definition for DEL L. **D + EL.** is the nokeypad definition for DEL EOL. CTRL/U (**DBL.** is its nokeypad definition) deletes the text from the character to the left of the cursor to the beginning of the line.

Example 1: Deletes the characters from the current cursor position to the end of the line, including the line terminator.

```
The committee will be composed of George, Mary, Larry,
Bill, and Peter.
```

```
The committee will be composed of George, Mary, Bill, and Peter.
```

Example 2: Alphabetizes the list by deleting the third line and using UND L to insert it in its proper location.

```
Anderson, Richard
Andrews, Harold
Andorsky, Thomas
Anthony, James
```

DEL L
UND L

(Move cursor to the **A** in **Andrews**.)

GOLD + DEL L / UND L

```
Anderson, Richard
Andorsky, Thomas
Andrews, Harold
Anthony, James
```

Related Commands: Line – DELETE
Nokeypad – D + NL (delete to next line)

Keypad Designations:

Description:

DEL W (delete word) deletes words or parts of words. When the cursor is at the beginning of the word, the entire word and the space following it are deleted. If the cursor is in the middle of the word, only the character that the cursor is on and those characters to the right of the cursor up to and including the following space(s) are deleted. The characters to the left of the cursor in that word remain in the text. If the word being deleted is at the end of a line, all characters up to, but not including, the line terminator are deleted.

The characters deleted by DEL W and **LINEFEED** (F13 on LK201 keyboards) are stored in the delete word buffer. Each time DEL W or LINEFEED is used, the contents of the delete word buffer are overwritten. Use **UND W** to restore or insert the contents of the delete word buffer into the text you are editing.

DEL W always deletes the cursor character and the remaining characters in the word to the right of the cursor. LINEFEED deletes the word or part of word to the left of the cursor. **DEW.** is the nokeypad definition for DEL W; **DBW.** is the nokeypad definition for LINEFEED.

Example 1: Deletes the word **Major** from the General's title.

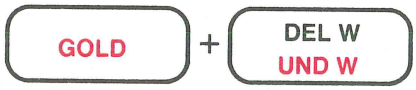
```
Major General George H. Mitchell
```

```
General George H. Mitchell
```

Example 2: Changes the order of the names by deleting the last name and then using UND W to insert it after the first name.

```
Smith, Brian
```

(Move the cursor to the end of the line and add a space after **Brian**.)




```
Brian Smith,
```

Related Commands: Line – DELETE
Nokeypad – DEW (delete to end of word)

Syntax:

```
DESEL
```

Description:

DESEL (deactivate select) cancels a select range after you have used the **SEL** (select) command. If you change your mind about a select range, issue the DESEL command. You can then continue with your editing work. If you want to change a select range that is currently active, use DESEL to cancel it and then SEL to mark the beginning of the new range.

DESEL also cancels a select range set by either the **SSEL** (search and select) or **TGSEL** (toggle select) command.

Example: Creates a select range of five lines. Then cancels that range, because the first line of the next address was included by mistake. Moves the cursor back one line and creates a four-line select range, using the minus sign (–) to reverse direction only for that command.

```
SEL5L

    Albuquerque, NM  87112

    Mr. Edward B. Sanders
    1749 General Lee Highway
    Richmond, VA  23235

    Dr. Jane Marantz-Connor
    0

DESEL

    Albuquerque, NM  87112

    Mr. Edward B. Sanders
    1749 General Lee Highway
    Richmond, VA  23235

    Dr. Jane Marantz-Connor
    0

BL SEL-4L

    Albuquerque, NM  87112

    Mr. Edward B. Sanders
    1749 General Lee Highway
    Richmond, VA  23235

    Dr. Jane Marantz-Connor
```

Related Commands: Keypad – RESET

DLWC (Default Lowercase) Command
Nokeypad

Syntax:

```
DLWC
```

Description:

The DLWC (default lowercase) command instructs EDT to change all uppercase letters to lowercase letters whenever you move the cursor during your EDT session. As the cursor moves through the buffer, every uppercase letter it passes over is changed to lowercase.

Since DLWC resets the default state, uppercase letters continue to be changed to lowercase each time you move the cursor. Use the **DMOV** (default move) command to reset EDT so that case is not affected by move operations. **DUPC** (default uppercase) causes EDT to change all lowercase letters that the cursor passes over to uppercase.

Example: Changes EDT's default to lowercase and then shows the effect by moving the cursor over four lines of text.

```
DATE:   06-APR-84
FROM:   Linda Westbrook
DEPT:   CED
NET:    QQQQ::WESTBROOK

DLWC 4L

date:   06-apr-84
from:   linda westbrook
dept:   ced
net:    qqqq::westbrook
```

Syntax:

```
DMOV
```

Description:

The DMOV (default move) command returns your editing session to EDT's default state after you have used either **DLWC** (default lowercase) or **DUPC** (default uppercase). With DLWC, EDT automatically changes the case of every uppercase letter to lowercase as the cursor moves through the text. With DUPC, all lowercase letters become uppercase as EDT moves through the buffer. For example, if you move from the current line to the end of the buffer with DUPC in effect, that entire portion of your buffer will contain only uppercase letters. DMOV sets things back to the normal state, where EDT does not alter the case of letters during move operations.

Example: Uses DLWC to change the default to lowercase and moves four words. Then restores the default to EDT's normal state – DMOV.

```
    The SOFTWARE SERVICES Training Proposal was issued last
    May for the current fiscal year.

DLWC 4W

    The software services training proposal was issued last
    May for the current fiscal year.

DMOV
```

DO Function
Keypad (LK201 only)

Keypad Designations:

Description:

DO processes searches and line editing commands in keypad mode. The period (.) is the nokeypad definition for DO. Although DO has the same definition as **ENTER**, you cannot use the DO key to enter a key definition with **CTRL/K**.

When you receive a prompt from EDT in keypad mode, you can use DO to send EDT the information you type in response to the prompt. The two preset EDT functions that have prompts are **COMMAND** and **FIND**.

To use COMMAND, press the GOLD and COMMAND keys. When EDT displays the **Command:** prompt, type the line mode command. Then press DO to send the command to EDT for processing.

To use FIND, press either the LK201 **Find** key or the GOLD and FIND keys on the numeric keypad. When EDT displays the **Search for:** prompt, type the search string. Then press DO to send the string to EDT so it can perform the search.

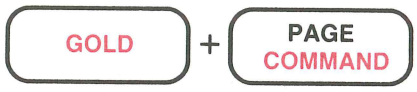
Example 1: Processes the FIND function using the LK201 Find key.

```
Search for:  Chicago
```

```
    meeting to be held in Chicago on April 9, 1984
```

Example 2: Processes the COMMAND function.

```
    meeting to be held in Chicago on April 9, 1984
```




```
Command:  SUBSTITUTE/Chicago/Boston/
```

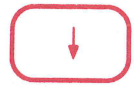
DO

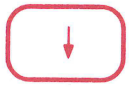
```
meeting to be held in Boston on April 9, 1984
```

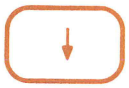
Related Commands: Line – RETURN
Nokeypad – RETURN

Down Arrow
Keypad
Nokeypad

Key Designation:







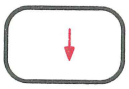
Description:

The Down Arrow key moves the cursor down one line toward the bottom of the buffer regardless of EDT's direction. **+V.** is the nokeypad definition for Down Arrow.

When you use the Down Arrow, EDT attempts to maintain the same vertical column as it moves the cursor from one line to the next. If there are not enough characters to fill out a line of text, the cursor moves to the end of that line. If you continue to use Down Arrow, the cursor will return to the same vertical column for all lines that have enough characters. However, once you press some other key, EDT cancels the column position for Down Arrow and resets it the next time you use the function.

Example: Moves the cursor from the end of the first line to the end of the last line.

```
Asuncion, Paraguay
Bogota, Colombia
Brasilia, Brazil
Buenos Aires, Argentina
Caracas, Venezuela
Georgetown, Guyana
```



```
Asuncion, Paraguay
Bogota, Colombia
```

```
Bogota, Colombia
Brasilia, Brazil
Buenos Aires, Argentina
```

```
Buenos Aires, Argentina
Caracas, Venezuela
Georgetown, Guyana
```

Syntax:

```
DUPC
```

Description:

The DUPC (default uppercase) command instructs EDT to change all lowercase letters to uppercase whenever you move the cursor during your EDT session. As the cursor moves through the buffer, every lowercase letter it passes over is changed to uppercase.

Since DUPC resets the default state, lowercase letters continue to be changed to uppercase each time you move the cursor. Use the **DMOV** (default move) command to reset EDT so that case is not affected by move operations. **DLWC** (default lowercase) causes EDT to change all uppercase letters that the cursor passes over to lowercase.

Example: Changes EDT's default to uppercase and then shows the effect by moving the cursor over four lines of text.

```
Karen Islington
Paul Ohlmeyer
William Jacobson
Sarah Rapf

DUPC 4L

KAREN ISLINGTON
PAUL OHLMEYER
WILLIAM JACOBSON
SARAH RAPF
```

/DUPLICATE Qualifier
Line

Syntax:

```
/DUPLICATE:n
```

Description:

/DUPLICATE is a qualifier used with the line mode **COPY** command. All EDT qualifiers must be preceded by slashes. The /DUPLICATE qualifier instructs EDT to copy the specified text **n** times above the location-2 specified in the COPY command. You must supply a value for **n** between 1 and 32767.

Example: Inserts five copies of line 3 above line 14.

```
*COPY 3 TO 14 /DUPLICATE:5
1 line copied 5 times

*TYPE 13 THRU 14

    13        1 pkg. #2 copper tubing
    13.1      1 pkg. #3 copper tubing
    13.2      1 pkg. #3 copper tubing
    13.3      1 pkg. #3 copper tubing
    13.4      1 pkg. #3 copper tubing
    13.5      1 pkg. #3 copper tubing
    14        1 pkg. #4 copper tubing
```

Keypad Designations:

Description:

ENTER processes searches, line editing commands, and key definitions in keypad mode. EDT generally uses the ENTER function to process keypad editing functions. The period (.) is the nokeypad definition for ENTER.

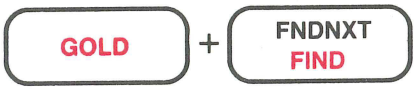
When you receive a prompt from EDT in keypad mode, use ENTER to send EDT the information you type in response to the prompt. The two preset EDT functions that have prompts are **COMMAND** and **FIND**.

To use COMMAND, press the GOLD and COMMAND keys. When EDT displays the **Command:** prompt, type the line mode command. Then press ENTER to send the command to EDT for processing.

To use FIND, press the GOLD and FIND keys. When EDT displays the **Search for:** prompt, type the search string. Then press ENTER to send the string to EDT so it can perform the search.

You are asked to press the ENTER key when you complete a keypad definition using **CTRL/K** in keypad mode. When the message **Now enter the definition terminated by ENTER** appears, type the definition and then press the ENTER key.

Example 1: Processes the FIND function.




```
Search for:  Chicago
```

```
    meeting to be held in Chicago on April 9, 1984
```

Example 2: Processes the COMMAND function.

```
meeting to be held in Chicago on April 9, 1984
```

GOLD + PAGE COMMAND

```
Command:  SUBSTITUTE/Chicago/Boston/
```

ENTER SUBS

```
meeting to be held in Boston on April 9, 1984
```

Example 3: Completes the processing of the CTRL/K key definition operation. The key definition shown in this example creates a keypad substitute function that prompts for both the search and the substitute strings.

CTRL/K

```
Press the key you wish to define
```

GOLD + S

```
Now enter the definition terminated by ENTER
S/?'Find: '/?' Substitute: '/.
```

ENTER SUBS

Example 4: Using the new substitute function created in Example 3, sends the strings to EDT after you type them.

```
the line entity.
```

GOLD + S

```
Find: line
```

ENTER SUBS

```
Find: line Substitute: paragraph
```

ENTER SUBS

```
the paragraph entity.
```

Related Commands: Line – RETURN
Nokeypad – RETURN

entity specifier
Nokeypad

Syntax:

```
entity
```

Description:

An **entity** is a group of contiguous characters that EDT recognizes as a unit. The entity specifier is used with the following nokeypad commands:

APPEND	FILL
CHGC	"move"
CHGL	R
CHGU	SEL
CUT	TADJ
D	TGSEL

The nokeypad mode entities are:

C	character	BR	beginning of range
L	line	ER	end of range
BL	beginning of line	SR	select range
EL	end of line	SEN	sentence
NL	next line	BSEN	beginning of sentence
PAGE	page	ESEN	end of sentence
BPAGE	beginning of page	"string"	string of characters
EPAGE	end of page	V	vertical
PAR	paragraph	W	word
BPAR	beginning of paragraph	BW	beginning of word
EPAR	end of paragraph	EW	end of word

See Table 5.1 in Chapter 5 for a more complete description of each entity.

Examples:

```
CHGCW
```

Changes the case of the letters in the next word.

```
APPEND2PAR
```

Deletes the next two paragraphs and places them at the end of the PASTE buffer.

```
DBR
```

Deletes to the beginning of the buffer.

```
3CUTSEN=EXTRA
```

Deletes the next three sentences and places them in the buffer named EXTRA.

Keypad Designations:

Description:

EOL (end of line) moves the cursor to the end of the current line if the direction is forward. If the current direction is backward, the cursor moves to the end of the previous line. If the cursor is already at the end of a line, EOL moves it to the end of the next or previous line depending on the current direction. Use **BACKSPACE** (F12 on LK201 keyboards) to move the cursor to the beginning of a line. **EL.** is the nokeypad definition of EOL.

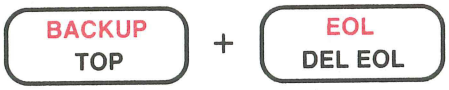
Example 1: Moves the cursor to the right to the nearest line terminator.

```
The me@ting is scheduled for Friday afternoon.
```

```
The meeting is scheduled for Friday afternoon.
```

Example 2: Moves the cursor to the left to the nearest line terminator, after setting the direction to backward.

```
3 typewriters
2 typing stands
```




```
3 typewriters
2 typing stands
```

Related Commands: Nokeypad – EL (end of line)

EX (Exit to Line Mode) Command
Nokeypad

Syntax:

```
EX
```

Description:

The EX (exit to line mode) command shifts your editing session from nokeypad mode to line mode. It does *not* cause an exit from EDT.

To exit from EDT when in nokeypad mode you can use EX to shift to line mode and then give the line mode **EXIT** or **QUIT** command. Use the nokeypad **QUIT** command to exit directly from nokeypad mode when you do not need to save your work. The nokeypad **EXT** (extend) command allows you to issue the line mode EXIT or QUIT command directly from nokeypad mode.

Example: Shifts from nokeypad mode to line mode.

```
          .
          .
          .
    Jennifer Smith      (401) 555-1111
    John W. Snyder      (401) 555-2222
          .
          .
          .

EX

*WRITE DELNAMES.DAT . THRU +1
DISK$USER:[SMITH]DELNAMES.DAT  2 lines
*FIND  =MAIN
*
```

Related Commands: Keypad – CTRL/Z
Line – CHANGE

Syntax:

```
EXIT [file-specification] [/SEQUENCE[:initial[:increment]]] [/SAVE]
```

Description:

The EXIT command ends your editing session. EDT creates an external file and copies the contents of the MAIN buffer into that file. When you type EXIT with no file specification, EDT uses the file specification information supplied in the EDIT/EDT command line. The file specification that you supply with the EXIT command supersedes any output file specification given in the EDIT/EDT command line. If the output file was only partially specified in the EDIT/EDT command line or EXIT command, EDT uses the input file name or file type in place of whichever item is missing.

If a directory name is included in the file specification, that directory must already exist and you must have access to it. When you use the EXIT command, EDT creates a file. However, EDT cannot create a directory.

The /**SEQUENCE** qualifier instructs EDT to retain the EDT line numbers for use in subsequent editing sessions. Only whole numbers (that is, line numbers with no decimals) can be used for the :initial and :increment specifiers. When /SEQUENCE is used with no specifiers, EDT attempts to retain the existing line numbers in so far as it can. But all line numbers with decimal fractions are adjusted to whole numbers. See the section in Chapter 7 on EDT line numbers for more information.

The /**SAVE** qualifier tells EDT to save the copy of the journal file in the current directory. The default journal file specification has the same file name as the input file. **.JOU** is the default file type. Unless you use the /SAVE qualifier in your EXIT command, EDT automatically deletes the journal file when you issue the EXIT command. The journal file contains all the keystrokes you made during the editing session. You can specify a different journal file name in the EDIT/EDT command line. See Chapters 2 and 7 for more information about EDT journal files.

The EXIT command tries to apply the attributes of the input file to the output file it creates. If these attributes have been violated by any editing work done during the session (for example, altering the length of a record in a fixed length record file on some systems), EDT might not be able to finish copying the MAIN buffer text into the output file. In these instances, EDT displays the error detected by the appropriate file service and discontinues processing the EXIT command. You can use the WRITE command to copy the text to an external file, but the attributes of the input file are no longer preserved. The WRITE command always creates an external file with EDT's default attributes. (Then use the QUIT command to exit from EDT after you have issued the WRITE command.) Information on EDT's default file attributes appears in the system appendixes.

Example 1: Shows the beginning and end of an EDT session. The input file is WKREP25.RNO and the output file is WKREP25.RNO;2. The system command DIRECTORY shows that both files exist in the directory after the EXIT command is used.

```
☞ EDT /EDT WKREP25.RNO

    1        .FILL.JUSTIFY

         .
         .
         .

*EXIT
DISK$USER:[SMITH:STATUS]WKREP25.RNO;2  63 lines

☞ DIRECTORY WKREP25.RNO
    WKREP25.RNO;1          WKREP25.RNO;2
```

Example 2: Shows the beginning and end of an EDT session. The input file is LETTER.RNO;4 and the output file is SANCHEZ.RNO;1. The system command DIRECTORY shows that both files exist in the directory after the EXIT command is used.

```
☞ EDT /EDT LETTER.RNO;4
    1        .FILL.JUSTIFY

         .
         .
         .

*EXIT SANCHEZ.RNO
DISK$USER:[MARSDEN.LETTERS]SANCHEZ.RNO;1 37 lines

☞ DIRECTORY
         .
         .
         .
    LETTER.RNO;4
         .
         .
         .
    SANCHEZ.RNO;1
```

Example 3: Shows the beginning and end of an EDT session. The input file is EOL.RNO and the output file is ENDOFLINE.RNO. The /SAVE qualifier causes the journal file to be saved so this editing work can be repeated with other files. Uses the DCL DIRECTORY command to show the journal file.

```
☞ EDIT /EDT EOL.RNO

    1       .FILL.JUSTIFY.LEFT MARGIN0.RIGHT MARGIN70
*SUBSTITUTE/GIN0/GIN10/
    1       .FILL.JUSTIFY.LEFT MARGIN10.RIGHT MARGIN70
1 substitution
*SUBSTITUTE/70/80/
    1       .FILL.JUSTIFY.LEFT MARGIN10.RIGHT MARGIN80
1 substitution
*INSERT ;.PAGE
    1       .FILL.JUSTIFY.LEFT MARGIN10.RIGHT MARGIN80

*EXIT ENDOFLINE.RNO /SAVE
DISK$USER:[SMITH.CHAPTER8]ENDOFLINE.RNO;1 54 lines

☞ DIRECTORY *.JOU
    EOL.JOU;1
```

EXT (Extend) Command
Nokeypad

Syntax:

```
EXT line mode command
```

Description:

The EXT (extend) command allows you to use a line mode command while still in nokeypad mode. The text following EXT must fit on a single line.

You can issue two or more line mode commands on the same line by separating the commands with semicolons (;). If you want to put nokeypad commands after a line mode command, use **CHANGE ;nokeypad-command(s)** as the last line mode command on the EXT command line. You can use an EDT macro with the EXT command just as you would any other line mode command.

Example 1: Enables you to enter several line mode SET commands on a single line directly from nokeypad mode.

```
EXT SET SEARCH EXACT ;SET WORD NODELIMITER ;SET SCREEN 100
```

Example 2: Enables you to use the line mode EXIT command directly from nokeypad mode.

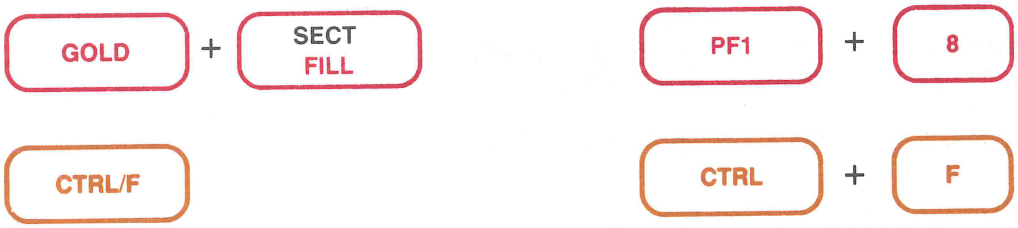
```
EXT EXIT

    DISK$USER:[SMITH:MEMOS]JAN1.RNO;6  34 lines
```

Example 3: Uses two line mode commands followed by a nokeypad command to reposition the cursor after the line mode commands are processed.

```
EXT COPY =SAVE 4 THRU 27 TO =TEST ;SUBSTITUTE/10/5/ WHOLE; CHANGE ;5L 3C
```

Related Commands: Keypad – COMMAND

Keypad Designations:



Description:

The FILL function is available on all VT100-type terminals. You must use CTRL/F on VT52 terminals to perform the FILL function.

FILL takes a select range of lines and reorganizes the text so that the maximum number of whole words can fit within the current line width. The default line width for EDT is the terminal width that the operating system passes to EDT. Use the line mode **SHOW SCREEN** command to find out the current screen/line width. The valid screen width values for screen mode editing are 80 and 132. (132 is only valid for VT100-type terminals with AVO – advanced video option.) If your screen width is set to 80, EDT will fill lines to column 79; if your screen width is 132, EDT will fill lines to column 131.

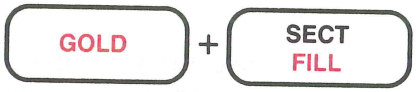
If you want to use a line length other than 80 or 132 for filling text, you must use the line mode **SET WRAP** command. The SET WRAP command also affects the line length that EDT uses for inserting text in keypad mode. EDT uses the SET SCREEN value to determine the line length for filling text only if SET NOWRAP (the default) is in effect. If SET WRAP is in effect, EDT always uses the wrap value, regardless of the SET SCREEN width. You can use the **SHOW WRAP** command to find out the current wrap value or setting.

The filling process considers a blank line to be a break between paragraphs. Even if there are spaces on the blank line, EDT fills the text up to the blank line and then resumes filling with the next line that contains text.

FILLSR. is the nokeypad definition for FILL on VT100-type terminals and for CTRL/F on VT52 terminals.

Example 1: Creates a select range of four lines and then reformats it using the current SET SCREEN width – 80 characters.

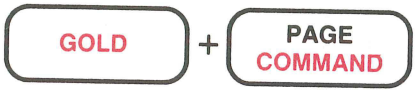
```
When the last line of the file is reached,
the computer reads the end of block
indicator
and displays the system prompt character.
```



```
When the last line of the file is reached, the computer reads the end of block
indicator and displays the system prompt character.
```

Example 2: First uses the line mode SET WRAP command to limit the line length to 30 characters. Then creates a select range of four lines and reformats the text.

```
When the last line of the file is reached,
the computer reads the end of block
indicator
and displays the system prompt character.
```




```
Command:  SET WRAP 30
```

SELECT RESET + LINE OPEN LINE + LINE OPEN LINE + LINE OPEN LINE + LINE OPEN LINE

GOLD + SECT FILL

GOLD + PAGE COMMAND

```
Command:  SET WRAP 30
```

ENTER SUBS

SELECT RESET + LINE OPEN LINE + LINE OPEN LINE + LINE OPEN LINE + LINE OPEN LINE

CTRL/F

```
When the last line of the file
is reached, the computer reads
the end of block indicator and
displays the system prompt
character.
```

Related Commands: Line – FILL
Nokeypad – FILL

FILL Command
Line

Syntax:

```
FILL [=buffer] [range]
```

Description:

The FILL command takes a select range of lines and reorganizes the text so that the maximum number of whole words can fit within the current line width. The default line width for EDT is the terminal width that the operating system passes to EDT. Use the line mode **SHOW SCREEN** command to find out the current screen/line width.

You can use either the **SET WRAP** or **SET SCREEN** line mode command to change the line width for filling. SET SCREEN changes the line width for all operations in your editing session. SET WRAP affects only filling. EDT uses the SET SCREEN width for filling if SET NOWRAP (the default) is in effect. If SET WRAP is in effect, EDT always uses that value, regardless of the SET SCREEN width. Both SET SCREEN and SET WRAP take the line length in characters as the command specifier. For example:

```
SET SCREEN 40
SET WRAP 60
```

Use the **SHOW WRAP** command to find the current wrap value or setting.

The fill process considers a blank line to be a break between paragraphs. Even if there are spaces on the blank line, EDT fills the text up to the blank line and then resumes filling with the next line that contains text.

If you do not specify a range or buffer with the FILL command, EDT assumes that a select range has been established in one of the screen modes. If no select range is active, EDT displays an error message.

If your SET SCREEN width is 80, EDT will fill lines to column 79; if your SET SCREEN width is 132, EDT will fill lines to column 131.

Example: Uses the SET WRAP command to limit the line length to 40 characters. Then reformats lines 10 through 15.

```
10        With regard to your letter of January 4th,
11        I am unable to satisfy your need for terminals
12        that scroll sideways.  To the best of my knowledge,
13        no such terminals are available on today's market.
14        We do, however,
15        have an editor named EDT with this capability.
```

```
*SET WRAP 40
*FILL 10 THRU 15

    10        With regard to your letter of January
    11        4th, I am unable to satisfy your need
    12        for terminals that scroll sideways.  To
    13        the best of my knowledge, no such
    14        terminals are available on today's
    15        market. We do, however, have an editor
    16        named EDT with this capability.
```

Related Commands: Keypad – FILL
Nokeypad – FILL

FILL Command
Nokeypad

Syntax:

```
[+|-][count]FILL[+|-][count]entity
```

Description:

The FILL command takes a select range of lines and reorganizes the text so that the maximum number of whole words can fit within the current line width. The default line width for EDT is the terminal width that the operating system passes to EDT. Use the line mode **SHOW SCREEN** command to find out the current screen/line width. The valid screen width values for screen mode editing are 80 and 132. (132 is only valid for VT100-type terminals with AVO – advanced video option.) If your screen width is set to 80, EDT will fill lines to column 79; if your screen width is 132, EDT will fill lines to column 131.

If you want to use a line length other than 80 or 132 for filling text, you must use the line mode **SET WRAP** command. The SET WRAP command also affects the line length that EDT uses for inserting text in keypad mode. EDT uses the SET SCREEN value to determine the line length for filling text only if SET NOWRAP (the default) is in effect. If SET WRAP is in effect, EDT always uses the wrap value, regardless of the SET SCREEN width. You can use the **SHOW WRAP** command to find out the current wrap value or setting.

The filling process considers a blank line to be a break between paragraphs. Even if there are spaces on the blank line, EDT fills the text up to the blank line and then resumes filling with the next line that contains text.

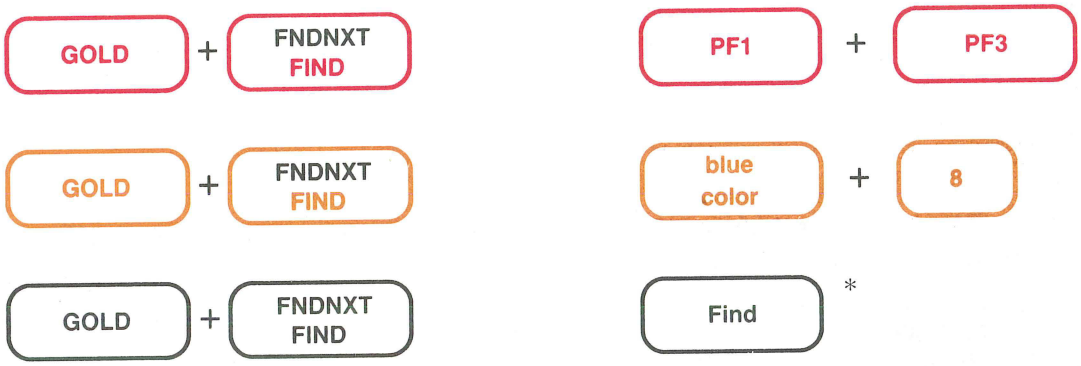
The valid entities for the FILL command are: L (line), SEN (sentence), PAR (paragraph), PAGE, and SR (select range).

Example: Uses the SET WRAP command to limit the line length to 60 characters. Then reformats the first six lines of text.

```
A computer is a tool that you can use to
process information and solve complicated problems.
It accepts large amounts of information,
called data,
performs various operations on that data,
and provides you with a result.

In order to process data,
there must be communication between you and the computer.

EXT SET WRAP 60
FILL6L

A computer is a tool that you can use to process information
and solve complicated problems. It accepts large amounts of
information, called data, performs various operations on
that data, and provides you with a result.

In order to process data,
there must be communication between you and the computer.
```

Related Commands: Keypad – FILL
Line – FILL

Keypad Designations:



*On LK201 keyboards, both the PF1/PF3 key sequence and the Find key have the same preset function.

Description:

FIND sets up a search procedure. When you press GOLD and then FIND, EDT displays the prompt **Search for:** at the bottom of the screen. Type the string you want to locate. Then push **ENTER** to process the search in the current direction.

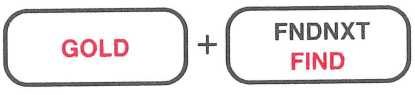
After you have typed in your search string, you can press **ADVANCE** instead of ENTER to search toward the end of the buffer or you can press **BACKUP** to search backward toward the top. The direction you use to process FIND becomes EDT's current direction.

EDT can perform searches in several ways. The default ways are GENERAL, BEGIN, and UNBOUNDED. GENERAL means that EDT ignores the case and diacritical marks of letters in performing searches. BEGIN means that EDT places the cursor on the first character of the search string. UNBOUNDED means that EDT performs the search in the portion of the buffer between the cursor position and the beginning or end of the buffer, depending on the direction of the search. Use the **SET SEARCH** command to change the way EDT performs searches. The **SHOW SEARCH** command tells you which search parameters are currently in effect.

^@?'Search for: '^@. is the nokeypad definition for FIND.

Example 1: Searches for the next occurrence of **Walker** in the current direction.

```
Wes Chandler
Hershel Walker
Wesley Walker
```




```
Search for:  Walker
```

ENTER
SUBS

```
Wes Chandler
Hershel Walker
Wesley Walker
```

Example 2: Using the same list of names with the cursor in its new position, searches for **Wes** in the backward direction.

GOLD + FNDNXT FIND

```
Search for:  Wes
```

BACKUP
TOP

```
Wes Chandler
Hershel Walker
Wesley Walker
```

Example 3: Using the same list of names with the cursor in its new position, searches for **Wes** in the forward direction.

GOLD + FNDNXT FIND

```
Search for:  Wes
```

ADVANCE
BOTTOM

```
Wes Chandler
Hershel Walker
Wesley Walker
```

Example 4: Using the same list of names with the cursor in its new position, searches for the next occurrence of **Walker** in the current direction.

GOLD + FNDNXT FIND

```
Search for:  Walker
```

ENTER SUBS

```
    Wes Chandler
    Hershel Walker
    Wesley Walker
```

Related Commands: Line – FIND
Nokeypad – "string"

FIND Command
Line

Syntax:

```
FIND [=buffer] [range]
```

Description:

The FIND command moves EDT to a new position in the current buffer or in the specified buffer. In line mode, once the specified line has been found, EDT signals with the line mode prompt (*). The new current line is not displayed at the terminal. (Use the **TYPE** command if you want to see the line.)

If you use the FIND command from keypad or nokeypad mode, the cursor moves to the first character of the specified line. If you use a string for range, the cursor moves to the character to the right of the search string.

When FIND is used with a buffer specifier but no range specifier, EDT positions itself at the first character of the specified buffer. When you use both buffer and range specifiers, EDT positions itself at the line you specified as the range in the specified buffer. Use a space to separate the buffer name from the range specifier if the range specifier begins with a letter or digit (for example, =PASTE(SP)1). Otherwise, EDT will interpret the range as part of the buffer name.

With the FIND command, you can search either forward or backward for a string in the current buffer. To search backward toward the beginning of the buffer, you must precede the search string with a minus sign (–). When you use a line number as the range specifier, FIND will locate the line regardless of direction.

If you use a string as the range specifier, you should be aware of how EDT searches for a matching string in the text. EDT performs searches in several ways. The default ways are GENERAL, BEGIN, and UNBOUNDED. GENERAL means that EDT ignores the case and diacritical marks of letters in performing searches. BEGIN means that EDT places the cursor on the first character of the search string. UNBOUNDED means that EDT performs the search on the portion of the buffer between the current position and the beginning or end of the buffer, depending on the direction of the search. Use the **SET SEARCH** command to change the way EDT performs searches. The **SHOW SEARCH** command tells you which search parameters are currently in effect. (BEGIN is only applicable in change modes.)

The FIND command is frequently used to move EDT from one buffer to another. When you type FIND =buffer name, EDT moves to the first line of the specified buffer. If that buffer does not already exist, EDT creates the buffer.

You can use the period range specifier (.) to return to the last line you were working on in a previously used buffer. For example, suppose 254.2 was the last line EDT was at in the MAIN buffer. When you finish your editing in the TEST buffer, type **FIND =MAIN.** to return to the MAIN buffer. EDT positions itself at line 254.2 in the MAIN buffer. (You can place the period immediately after the buffer name without confusing EDT, because periods are not valid characters in buffer names.)

Example 1: Moves EDT to line 15, but prints nothing. The new location is correct for the SUBSTITUTE command.

```
   11       In regard to your letter of the 24th, we are planning

*FIND 15
*SUBSTITUTE/December/January/

   15       I will be flying to Denver on January 15th, arriving at
1 substitution
```

Example 2: Moves EDT to the next line containing the string **Bob** but prints nothing. The new location is correct for the SUBSTITUTE command.

```
*FIND "Bob"
*SUBSTITUTE/Bob/Harry/

   21       with Harry at the next Board of Directors Meeting.
1 substitution
```

Example 3: Moves to the end of the buffer so that you can insert a new line at the end of the text.

```
*FIND END
*INSERT; cc:  John C. Fredericks
[EOB]
*
```

Example 4: Moves to the buffer named LIST but prints nothing. The TYPE command simply verifies the new location.

```
*FIND =LIST

*TYPE

   1       Distribution List for File Copies
```

Related Commands: Keypad – FIND
Nokeypad – "string"

FNDNXT (Find Next) Command
Keypad

Keypad Designations:

Description:

After a search string has been established by **FIND**, you can use FNDNXT (find next) to locate the next occurrence of that string. The direction for FNDNXT is always the current EDT direction. You can change directions without affecting the search string.

The search string established by FIND remains in effect until you use FIND again or use some other EDT function that overwrites the contents of the search buffer.

EDT can perform searches in several ways. The default ways are GENERAL, BEGIN, and UNBOUNDED. GENERAL means that EDT ignores the case and diacritical marks of letters in performing searches. BEGIN means that EDT places the cursor on the first character of the search string. UNBOUNDED means that EDT performs the search on the portion of the buffer between the cursor position and the beginning or end of the buffer, depending on the direction of the search. Use the **SET SEARCH** command to change the way EDT performs searches. The **SHOW SEARCH** command tells you which search parameters are currently in effect.

" ". is the nokeypad definition for FNDNXT.

Example 1: First uses FIND to load the search buffer. Then uses FNDNXT to locate the fourth occurrence of the string **wi** in the list.

```
Dwight Evans
Willie Randolph
Willie Wilson
Dave Winfield
```

```
Search for:  wi
```

```
Dwight Evans
Willie Randolph
Willie Wilson
Dave Winfield
```

FNDNXT FIND + FNDNXT FIND + FNDNXT FIND

```
Dwight Evans
Willie Randolph
Willie Wilson
Dave Winfield
```

Example 2: With the current search string still **wi**, but the direction changed to backward, locates the string at the beginning of the third line.

BACKUP TOP

FNDNXT FIND

```
Dwight Evans
Willie Randolph
Willie Wilson
Dave Winfield
```

Related Commands: Line – FIND " "
Nokeypad " "

GOLD Function
Keypad

Keypad Designations:

Description:

GOLD works together with other keypad and keyboard keys to perform various editing functions. GOLD is like the SHIFT key in that it does nothing by itself.

When used with a keypad key, GOLD causes EDT to perform that key's alternate function. For example, to use the COMMAND function, you must first press GOLD and then the 7 key on the keypad. If you do not press GOLD, EDT performs the PAGE function. Using EDT's key definition facility, you can redefine any GOLD/keypad sequence to perform a different function during your EDT session.

The define key feature allows you to designate a GOLD/keyboard key sequence to perform a keypad editing function for the duration of your editing session. You can also use GOLD in combination with a CTRL/character sequence and with the DELETE key to define new keypad mode functions. GOLD/FUNCTION key sequences can be defined on terminals that have LK201 keyboards. See Chapter 7 for information on how to define and redefine function keys for keypad editing.

GOLD can be used with keyboard number keys to designate the number of times for EDT to repeat a keypad editing function. First press GOLD, next the keyboard number key(s), and then the keypad function key(s) that you want EDT to repeat. When EDT's direction is set to forward, you can use GOLD followed by a minus sign (–) to change EDT's direction to backward temporarily. This feature allows you to process a function that depends on EDT's current direction, in the backward direction, without having to reset EDT's direction. For example, you can use GOLD/–2 with **WORD** to have the cursor backup two words without changing EDT's direction. The maximum number of times you can repeat a function with the GOLD/repeat feature is 32767.

When you use the **SPECINS** function, you first press GOLD, then the keyboard digits for the decimal equivalent value of the character you want to insert. Then press GOLD again – this time to access the alternate function on the keypad function key – and finally the SPECINS key.

GOLD is the nokeypad definition for GOLD. Note that there is no period at the end of the definition. This is because GOLD is not a nokeypad command. You must use the line mode DEFINE KEY command to redefine a key that has GOLD as its definition.

Example 1: Causes EDT to process FIND, not FNDNXT.

GOLD + FNDNXT / FIND

Example 2: Causes EDT to perform the PASTE function, not CUT.

GOLD + CUT / PASTE

Example 3: Using CTRL/K, you define the key sequence GOLD/E to locate a semicolon (;) and delete to the end of line.

CTRL/K

```
Press the key you wish to define
```

GOLD + E

```
Now enter the definition terminated by ENTER
";"D+EL.
```

ENTER / SUBS

Example 4: Causes EDT to repeat the period (.) 60 times, thus creating a line of 60 dots.

GOLD + 6 + 0 + .

```
.............................................................
```

Example 5: Using the minus sign (–) as well as a repeat count of two, causes EDT to change the case of the two letters preceding the cursor.

```
New vt100 terminals are now available.
```

GOLD + – + 2 + GOLD + WORD / CHNGCASE

```
New VT100 terminals are now available.
```

Example 6: With SPECINS, inserts the form feed character into the text.

```
as of the last day in August.
□
The next item on the agenda will be to clarify the
```

```
as of the last day in August.
<FF>□
The next item on the agenda will be to clarify the
```

Keypad Designations:

*On LK201 keyboards, both the PF2 key and the Help key have the same preset function.

Description:

HELP provides information on EDT's preset keypad and control functions. Using HELP puts you in touch with EDT's HELP facility; it has no effect on your editing session. When you exit from HELP, the screen is redrawn exactly as it was before you pressed HELP and the cursor is in the same position as before.

When you press HELP, EDT displays a diagram of the keypad functions and a list of preset control key functions. For help on a particular keypad function key, press the appropriate keypad key. For information on a GOLD/keypad sequence, press only the keypad key. Information for both the primary and alternate functions of that key will be displayed. For information on a control key sequence, press both the CTRL and keyboard key after you are in the keypad HELP facility. For help on a GOLD/keyboard key sequence, press only the keyboard key; do not press GOLD.

To exit from HELP, press the spacebar. Figures 3-5 and 3-6 in Chapter 3 show the HELP diagrams for VT100-type and VT52 terminals.

If you have access to more than one HELP file, use the **SET HELP** command to change HELP files. The **SHOW HELP** command prints the name of the HELP file that is currently available for your editing session.

To define another key to perform the HELP function, use the nokeypad **HELP** command. **HELP.** is the nokeypad definition for HELP.

Related Commands: Line – HELP
[Nokeypad – HELP]

HELP Command
Line

Syntax:

```
HELP [topic [subtopic [subsubtopic] ] ]
```

Description:

The HELP command puts you in touch with EDT's HELP facility. When you type HELP by itself, EDT prints general information and instructions on how to get help for a specific topic. To get information on subtopics, you must type the name of the topic and then the name of the subtopic, for example, HELP EXIT /SAVE. Using the HELP facility has no effect on your editing session.

You can access the HELP file any time you see the line mode asterisk prompt (*). You can also access the information directly from keypad mode by using the **COMMAND** function or from nokeypad mode by using the **EXT** (extend) command.

HELP for nokeypad commands is available through the line mode HELP command. Type HELP CHANGE for information on nokeypad syntax and the list of subtopics. To get information on nokeypad commands, type HELP CHANGE SUBCOMMANDS.

If you have access to more than one HELP file, use the **SET HELP** command to change HELP files. The **SHOW HELP** command prints the name of the HELP file that is currently available for your editing session.

Examples:

```
*HELP JOURNAL
```
Prints information on EDT's journal facility.

```
*HELP CHANGE HARDCOPY
```
Prints information on hardcopy change mode.

```
*HELP DEFINE KEY VT100
```
Prints the keypad key number chart for VT100 terminals. These key numbers are used with the DEFINE KEY and SHOW KEY commands.

```
*HELP RESEQUENCE /SEQUENCE
```
Prints information on the /SEQUENCE qualifier when used with the RESEQUENCE command.

```
*HELP CHANGE SUBCOMMANDS
```
Prints general information on nokeypad commands.

```
*HELP CHANGE SUBCOMMANDS TADJ
```
Prints information on the nokeypad command TADJ.

Related Commands: Keypad – HELP

Syntax:

```
HELP
```

Description:

The nokeypad HELP command is used exclusively for defining a different key or key sequence in keypad mode to carry out the keypad HELP function. **HELP.** is the preset definition of keypad **HELP** key.

The nokeypad HELP command has no effect on your nokeypad editing session. To get help information on nokeypad commands, you must use the line mode **HELP** command:

```
HELP CHANGE [subtopic [subtopic] ]
```

The **subtopic** specifier in the HELP CHANGE command line refers to a nokeypad help topic. If you do not include any subtopics, EDT prints general information about nokeypad command syntax and a list of the available subtopics. Information on nokeypad entities appears under HELP CHANGE ENTITIES. Information on nokeypad commands appears under HELP CHANGE SUBCOMMANDS.

Example 1: Defines the key sequence GOLD/H to have the HELP function in keypad mode.

```
*DEFINE KEY GOLD H AS "HELP."
```

Example 2: Shows how to get three different types of information on nokeypad mode using the EXT (extend) command and the line mode HELP command.

```
EXT HELP CHANGE SUBCOMMANDS
```

Prints general information about nokeypad commands.

```
EXT HELP CHANGE SUBCOMMANDS S
```

Prints information about the nokeypad S (substitute) command.

```
EXT HELP CHANGE ENTITIES
```

Prints information about nokeypad entities.

I (Insert) Command
Nokeypad

Syntax:

```
I (RETURN) text (CTRL/Z)
Itext^Z
```

Description:

The I (insert) command shifts EDT to the insert state so you can insert new text in the current buffer. **CTRL/Z** signals EDT to exit from the insert state.

The I command has two forms. The first can be used for inserting any amount of text from a single character or word to several paragraphs or pages. The second is limited to text that fits on a single line and does not include a line terminator.

To insert any length of text, type I and press RETURN. The I remains on the command line while the cursor returns to its former position on the screen. Now type in the text you want to insert. When you have finished entering the new text, press CTRL/Z to exit from the insert state. The new text is now part of the current buffer and the I disappears from the command line to show that you are no longer in the insert state.

To insert less than a line of text, type the I command and then follow it immediately with the text you want to insert. (EDT interprets a space after the I as part of the text to be inserted.) Then type ^Z or press CTRL/Z to signal EDT that you have finished your inserted text. Finally, press RETURN to have EDT insert the text to the left of the cursor.

You can use the DELETE key to edit the text while you are inserting it. Or, you can wait until you finish typing the new text, and then use any nokeypad editing command to make the necessary changes.

Example 1: Adds two lines of text to the current buffer.

```
    Have the following materials ready:

         interim report
         planning budget
         
I (RETURN)
    staffing projections
    space requirements
(CTRL/Z)

    Have the following materials ready:

         interim report
         planning budget
         staffing projections
         space requirements
```

Example 2: Adds **Wednesday,**(SP) to the middle of the existing line.

```
    Before the meeting on(J)anuary 16th, please have the
IWednesday, ^Z
    Before the meeting on Wednesday, (J)anuary 16th, please have the
```

Related Commands: Line – INSERT

INCLUDE Command
Line

Syntax:

```
INCLUDE file-specification [=buffer] [range]
```

Description:

The INCLUDE command copies external files into an EDT text buffer. In line mode, EDT displays its asterisk prompt (*) when the INCLUDE command has finished copying the file. If you use the INCLUDE command from keypad or nokeypad mode, the included text appears on the screen.

The file specification is the name of the file you want to include in the specified text buffer. If the file name is omitted, EDT assumes the same name as the input file. Similarly, if the file type is omitted, EDT assumes the same file type as the input file. However, you must include some element of the file specification in the command line.

If no buffer is specified, the file is added to the current buffer. The range specifier refers to a single line. EDT puts the new text above the range line. If you omit the range specifier, EDT puts the text above the current line or at the top of the specified buffer. EDT assigns appropriate numbers to the new lines that are added to the buffer. If there are any sequence numbers stored with the included file, EDT ignores them.

Examples:

```
*INCLUDE LIST.DAT
```
EDT places a copy of the file LIST.DAT in the current buffer immediately above the current line.

```
*INCLUDE ADDR.DAT =ADDRESS
```
EDT places a copy of the file ADDR.DAT at the top of the buffer ADDRESS. If no buffer named ADDRESS exists, EDT creates one.

```
*INCLUDE [JACKSON]MEMO35.RNO 12
```
EDT places a copy of the file MEMO35.RNO from the directory JACKSON in the current buffer immediately above line 12.

```
*INCLUDE EXTRAPAR.X =MAIN 58
```
EDT places a copy of the file EXTRAPAR.X in the main buffer immediately above line 58.

Syntax:

```
INSERT [=buffer] [range] ;line-to-be-inserted
INSERT [=buffer] [range] (RETURN) text (CTRL/Z)
```

Description:

The INSERT command is used to add text to the current buffer or another buffer, if one is specified. Text is always inserted above the line specified by range, above the current line if you omit the range specifier, or at the top of the specified buffer if only a buffer name is supplied.

To insert a single line of text, you can type INSERT, followed optionally by a buffer and/or range specifier, then a semicolon (;), and finally the text to be inserted. You must press RETURN to process the one-line insert. If the semicolon is omitted, EDT prints an error message. To see the line you have just inserted you can use the **TYPE** command with –1 for the range specifier.

To insert one or more lines, type INSERT, followed optionally by a buffer and/or range specifier, and press RETURN. EDT is now in the insert state. You can type as many lines as you want. EDT assigns ascending line numbers to the new lines. When you have finished typing the text, press **CTRL/Z** to exit from the insert state.

If you use a buffer specifier with your INSERT command, EDT moves to the named buffer and remains there after you have finished inserting the text. You must use the **FIND** command to return to the buffer where you were previously editing.

Only the **INSERT ;line-to-be-entered** form of the command can be used in startup command files or EDT macros. To insert several lines of text in a startup command file or EDT macro, put **INSERT**(SP); before each text line.

Example 1: Adds the telephone number line after the last line of the address. Uses TYPE to show the newly inserted line.

```
     2        South Bend, IN  46628

*INSERT 3 ;(219) 555-1234

     3

*TYPE -1

     2.1      (219) 555-1234
```

Example 2: Moves to the buffer ADDRESS and inserts an address. Then uses FIND to return to the MAIN buffer.

```
*INSERT =ADDRESS (RETURN)
          Dr. James Grogan
          Charlotte Professional Building
          Suite 12A
          Charlotte, NC  28202
          (CTRL/Z)
[EOB]

*FIND =MAIN
```

Related Commands: Nokeypad – I

Syntax:

```
KS
```

Description:

The KS (KED substitute) command is used after the nokeypad **PASTE** command to modify the position of the cursor at the completion of the PASTE command. KS can only be used directly after the PASTE command. Do not put a space between the command words PASTE and KS.

Normally after you issue a PASTE command, the cursor is located one character to the right of the inserted text, regardless of EDT's current direction. When you put KS after the PASTE command, the cursor position changes and becomes dependent on EDT's current direction. If the direction is forward, the cursor is on the last character of the inserted text. If the direction is backward, the cursor is positioned on the first character of the inserted text.

The KS command enables you to locate a search string that includes the character immediately after or before the inserted text.

The nokeypad definition for the keypad mode **SUBS** function includes the KS command. Notice the KS command following directly after the PASTE command.

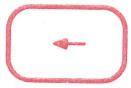
```
(CUTSR=DELETE PASTEKS"").
```

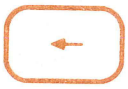
Example: Assumes that the text **Atlanta, GA,** is in the PASTE buffer. First resets the direction to backward. Then inserts the contents of the PASTE buffer and positions the cursor on the first character of the inserted text.

```
following locations:  Oakland, CA, Augusta, ME,
Billings, MT, Knoxville, TN.

BACK PASTEKS

following locations:  Oakland, CA, Augusta, ME,
Billings, MT, Atlanta, GA, Knoxville, TN.
```

Left Arrow
Keypad
Nokeypad

Keypad Designations:





Description:

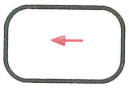
Left Arrow moves the cursor one character to the left, regardless of EDT's direction. –C. is the nokeypad definition for Left Arrow.

If the cursor is at the first character position of a line, pressing Left Arrow moves the cursor to the line terminator of the previous line.

Example: Moves the cursor to the left, first to the beginning of the second line and then to the end of the first line.

```
July 1, 1984
Oct@ber 1, 1984
```

```
July 1, 1984
@ctober 1, 1984
```



```
July 1, 1984@
October 1, 1984
```

Keypad Designations:

Description:

LINE moves the cursor to the beginning of the next line if the direction is forward or to the beginning of the current line if the direction is backward. If the cursor is at the beginning of a line and the direction is backward, the cursor moves to the beginning of the previous line. **L.** is the nokeypad definition for LINE.

Example 1: Moves the cursor from the middle of the second line to the beginning of the third line.

```
Software Documentation
Software Development
Production Groups
```

```
Software Documentation
Software Development
Production Groups
```

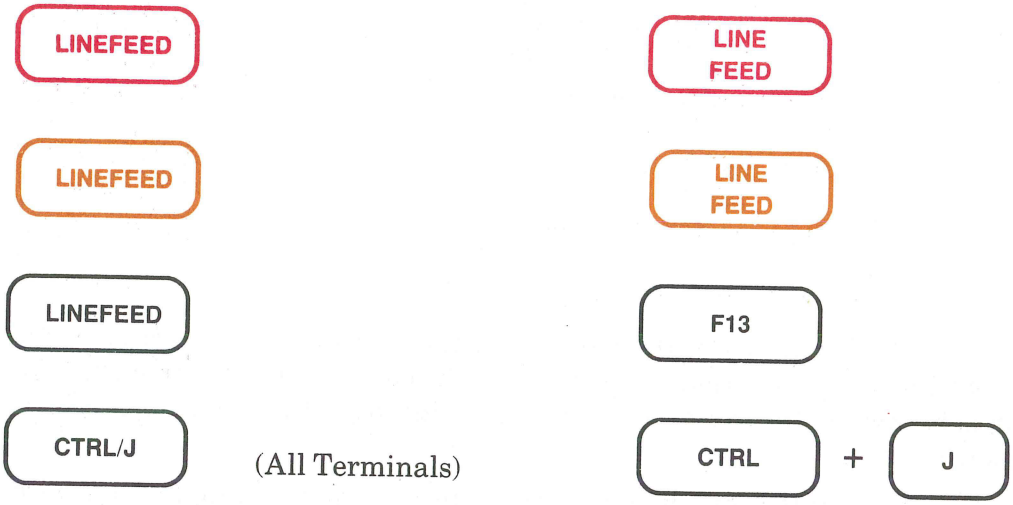
Example 2: Using the same text and original cursor position, reverses the direction and moves the cursor to the beginning of the first line.

```
Software Documentation
Software Development
Production Groups
```

Related Commands: Nokeypad – L (line)

LINEFEED Function CTRL/J
Keypad

Key Designations:



Description:

LINEFEED deletes the word or characters in a word to the left of the cursor up to the beginning of the previous word. It is similar to **DEL W**, which deletes the word or characters in a word to the right of the cursor up to the beginning of the next word. **DBW.** is the nokeypad definition for LINEFEED.

If the cursor is on a space when LINEFEED is pressed, the word preceding the space is deleted, usually leaving two spaces in a row. If the cursor is at the end or in the middle of a word, all characters in that word to the left of the cursor are deleted. The letter that the cursor is on remains in the text.

When the cursor is at the beginning of a word, the preceding word and space are deleted by LINEFEED. If the cursor is at the beginning of a line, LINEFEED deletes the preceding line terminator.

All characters deleted by LINEFEED are stored in the delete word buffer. Each time DEL W or LINEFEED is used, the contents of the delete word buffer are overwritten. Use **UND W** to insert or restore the contents of the delete word buffer in your text.

The LINEFEED key and CTRL/J always have the same preset function in EDT. When you redefine the LINEFEED key, you redefine CTRL/J (except for terminals with LK201 keyboards when they are operating in VT200 mode.) To redefine the LINEFEED key using the line mode **DEFINE KEY** command, type DEFINE KEY CONTROL J. To find out what the definition of the LINEFEED key is, type **SHOW KEY** CONTROL J. For terminals with LK201 keyboards, use DEFINE KEY FUNCTION 25 and SHOW KEY FUNCTION 25 for the F13 key.

Example: Deletes the characters to the left of the cursor up to the next word boundary.

```
This guide describes the VT100 terminal.
```

```
This guide describes the VT100 .
```

Related Commands: Nokeypad – DBW (delete to beginning of word)

MOVE Command
Line

Syntax:

```
MOVE [=buffer-1] [range-1] TO [=buffer-2] [range-2] [/QUERY]
```

Description:

The MOVE command moves the line(s) specified by location-1 (buffer-1 and/or range-1) to location-2 (buffer-2 and/or range-2). The lines specified in location-1 are deleted from their original position and inserted at the new location. Use the **COPY** command if you want to have the text appear in both places.

If location-1 is omitted, EDT moves the current line. When location-2 is omitted, EDT inserts the text above the current line. Whenever you omit the buffer specifier, EDT assumes the current buffer. If buffer-1 is specified without a range-1 specifier, the entire contents of buffer-1 are moved to location-2. If buffer-2 is specified without a range-2 specifier, the text is inserted at the beginning of buffer-2.

Range-1 can refer to one or more lines in the buffer, but range-2 is limited to a single line in the current or specified buffer. Remember to leave spaces between the buffer name and any range specifier if one is used. Otherwise, EDT might misinterpret the buffer name.

If you do not include a buffer-2 specifier, then EDT completes the MOVE command in the same buffer from which you issued the command. When you do specify buffer-2, that buffer becomes the current buffer.

The /**QUERY** qualifier allows you to make decisions during the MOVE operation. EDT displays each line in range-1 and prompts you with a question mark (?) so that you can decide whether or not you want to move that line. The four valid responses are: Y (YES), N (NO), A (ALL), and Q (QUIT).

Examples:

```
*MOVE 5 TO 15
```
Moves line 5 in the current buffer to be above line 15, also in the current buffer.

```
*MOVE =PAGE7 TO 198
```
Moves the entire contents of the buffer PAGE7 to be above line 198 in the current buffer.

```
*MOVE =ADDRESS 1 THRU 5 TO
```
Moves lines 1 through 5 in the buffer ADDRESS to be above the current line in the current buffer.

```
*MOVE 38 THRU 94 TO =SECTION1
```
Moves lines 38 through 94 in the current buffer to be at the beginning of the buffer SECTION1. If no buffer named SECTION1 exists, EDT creates it.

```
*MOVE TO =MAIN 26
```
Moves the current line in the current buffer to be above line 26 in the MAIN buffer.

Related Commands: Keypad – CUT [+ "move"] + PASTE
Nokeypad – CUT [+ "move"] + PASTE

Syntax:

```
[+|-][count]entity
```

Description:

The "move" command moves the cursor within the current buffer. The **entity** specifier is the only required element in the "move" command. There is no command word or abbreviation.

You can put several "move" commands together. For example, **L 3W –4C** moves the cursor down to the beginning of the next line, then three words to the right, and finally four characters to the left. In cases where you want to repeat several "move" commands, you can enclose the entities in parentheses: 8(EPAR +C).

Example 1: Moves the cursor to the beginning of the next line.

```
Only a few computer programs work successfully the first
time they are tried.  Programmers are expected to make
mistakes.  The mark of a good programmer is how quickly
he or she can find and correct the errors that occur.

L

time they are tried.  Programmers are expected to make
```

Example 2: Using the same text and the new cursor position, moves the cursor to the string **programmer**.

```
"programmer"

mistakes.  The mark of a good programmer is how quickly
```

Example 3: Using the same text and the new cursor position, moves the cursor three words to the right.

```
3W

mistakes.  The mark of a good programmer is how quickly
```

Example 4: Using the same text and the new cursor position, moves the cursor to the end of the last line.

```
2EL

he or she can find and correct the errors that occur.
```

Related Commands: Keypad – FIND, CHAR, WORD, LINE, PAGE
Line – FIND

NEXT Command
Line

Syntax:

```
[SUBSTITUTE] NEXT[/[string-1]/string-2/]
```

Description:

See SUBSTITUTE NEXT.

Syntax:

```
/NOTYPE
```

Description:

The /NOTYPE qualifier, which is used only with the line mode **SUBSTITUTE** command, instructs EDT not to display the lines in which substitutions have been made. EDT displays only the message giving the number of substitutions. Always precede an EDT qualifier with the slash.

Examples: Instructs EDT not to display the lines in which substitutions occurred.

```
*SUBSTITUTE/chairman/Chairperson/ WHOLE /NOTYPE
14 substitutions

*SUBSTITUTE*input/output*I/O* . THRU +20 /NOTYPE
7 substitutions
```

<null> Command Line

Syntax:

```
[=buffer] [[%]range]
```

Description:

The <null> command causes EDT to display the specified text. This command consists only of optional specifiers; there is no command word to type. If you omit both the buffer and range specifiers, the command consists of simply pressing the RETURN key in response to the asterisk prompt. In that case, EDT displays the next line in the current buffer.

The effect of the <null> command is the same as the **TYPE** command. For this reason the <null> command is referred to as the "implied TYPE" command.

When you include a buffer specifier, EDT moves to that buffer. If you do not specify a range, EDT displays every line in the buffer. Otherwise, EDT displays only the lines specified by range. When you move from one buffer to another, you can return to the same position in a previous buffer by using the period specifier (.) after the buffer name. For example, the command **=MAIN .** moves EDT back to the MAIN buffer on line 37.21 if 37.21 was the most recent current line in MAIN.

After EDT displays the specified lines, it positions itself at the first line of the range or buffer, not at the end.

When you specify certain ranges, there are some differences between the TYPE command and the <null> command. Range specifiers that begin with letters – ALL, BEGIN, END, LAST, and REST – must be preceded by a percent sign (%) when they are the first element of a <null> command. If the percent sign is omitted, EDT tries to interpret the word as a command or macro name and returns an error message.

The <null> command takes no qualifiers. You can use the **/BRIEF** and **/STAY** qualifiers with the TYPE command.

Example 1: Causes EDT to display lines 2 through 4.

```
1       This is the first line in the MAIN buffer.
2       This is the second line in the MAIN buffer.
3       This is the third line in the MAIN buffer.
4       This is the fourth line in the MAIN buffer.
5       This is the fifth line in the MAIN buffer.
6       This is the sixth line in the MAIN buffer.

*2 THRU 4

2       This is the second line in the MAIN buffer.
3       This is the third line in the MAIN buffer.
4       This is the fourth line in the MAIN buffer.
```

Example 2: Causes EDT to display the line below the current line.

```
*+1
     3        This is the third line in the MAIN buffer.
```

Example 3: Causes EDT to display the first line of the buffer.

```
*%BEGIN
     1        This is the first line in the MAIN buffer.
```

Example 4: Causes EDT to display the third line after the current line.

```
*+3
     4        This is the fourth line in the MAIN buffer.
```

Example 5: Causes EDT to display the next line after the current line when no specifier is supplied.

```
*(RETURN)
     5        This is the fifth line in the MAIN buffer.
```

Example 6: Causes EDT to display the line two lines above the current line.

```
*-2
     3        This is the third line in the MAIN buffer.
```

Example 7: Causes EDT to display all remaining lines in the buffer starting with the current line.

```
*%REST
     3        This is the third line in the MAIN buffer.
     4        This is the fourth line in the MAIN buffer.
     5        This is the fifth line in the MAIN buffer.
     6        This is the sixth line in the MAIN buffer.
 [EOB]
```

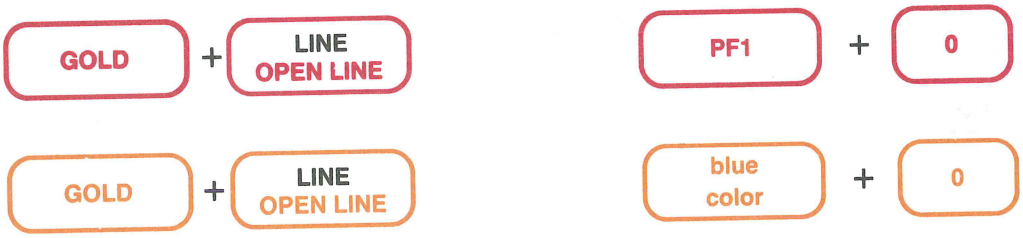
Example 8: Moves to the PASTE buffer and causes EDT to display the entire contents of that buffer.

```
*=PASTE

     1        This is the first line in the PASTE buffer.
     2        This is the last line in the PASTE buffer.
 [EOB]
```

Example 9: Moves to the buffer MEMO and displays lines 17 through 42.

```
*=MEMO 17 THRU 42

    17        after the product has been thoroughly tested.
    18
          .
          .
          .
    41        Please notify me of any changes in your schedule
    42        as soon as you become aware of them.
```

Keypad Designations:



Description:

OPEN LINE inserts a line terminator in the text you are editing at the current cursor position and makes the line terminator the new cursor character. If the cursor is initially at the beginning of a line, the text on that line is moved down so that the cursor is on the blank line.

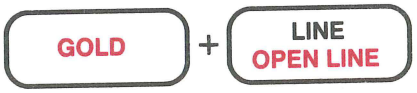
When the cursor is in the middle of a line, the text to the right of the cursor and the cursor character itself move to a new line. The cursor is now on the line terminator that OPEN LINE inserts. When the cursor is at the end of a line, a line terminator is added, creating a blank line below the current line.

RETURN and **CTRL/M** also insert line terminators in your text. However, neither of these functions moves the cursor to the inserted line terminator. The cursor remains on the same character.

(**^M–C**). is the nokeypad definition for OPEN LINE. **^M** is the definition of RETURN.

Example 1: Inserts a blank line between the first and second line.

```
Meeting Report
On Wednesday, I attended a seminar on the advanced
capabilities of the EDT Editor.  The topics covered
included startup command files, defining and
redefining keys, defining and using macros.
The presentation was very good.
```




```
Meeting Report

On Wednesday, I attended a seminar on the advanced
capabilities of the EDT Editor.  The topics covered
included startup command files, defining and
redefining keys, defining and using macros.
The presentation was very good.
```

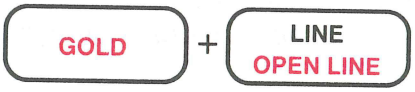
Example 2: Using the same text, inserts a blank line above the last line.

(Move the cursor to the beginning of the last line.)

```
Meeting Report

On Wednesday, I attended a seminar on the advanced
capabilities of the EDT Editor.  The topics covered
included startup command files, defining and
redefining keys, defining and using macros.
▯
The presentation was very good.
```

Example 3: Using the same text, inserts a line terminator that moves the words **The topics covered** to a new line.

(Move the cursor to the **T** in **The** in the middle of the second line.)




```
Meeting Report

On Wednesday, I attended a seminar on the advanced
capabilities of the EDT Editor.  ▯
The topics covered
included startup command files, defining and
redefining keys, defining and using macros.

The presentation was very good.
```

Related Commands: Line – INSERT RETURN
Nokeypad – I RETURN

Keypad Designations:

Description:

PAGE moves the cursor to a position at the right of the next page marker in your text. The cursor is always located after the page marker, but the direction that EDT moves to find the page marker depends on the current direction. In order to use PAGE, the text you are editing must have PAGE boundary markers. The default page marker is the form feed character (CTRL/L, decimal value 12, displayed by EDT as <FF>.)

PAGETOP. is the nokeypad definition for PAGE. This means that EDT moves the page marker line to the top of the screen if there are more than 22 lines between the page marker and the end of the buffer.

If you have no page markers in your buffer, the PAGE entity is the same as the whole buffer. When EDT's direction is forward, PAGE moves the cursor to the end of buffer ([EOB]) mark. If the current direction is backward, PAGE moves the cursor to the beginning of the buffer.

You can use the **SET ENTITY** PAGE command to define any string of characters as the page marker for the duration of your editing session. The marker can be either a single character that you insert in the text, such as an exclamation point (!), or a series of characters, such as a RUNOFF header level (.HL1).

If you are using the default page marker, you can use **SET TEXT** PAGE to have EDT display some other text in place of the <FF> page marker for the duration of your EDT session.

Example 1: Moves the cursor to the next page marker in the text.

```
and will be forthcoming as soon as we receive your payment.
<FF>
If you are not completely satisfied with this product, please
notify us immediately so that we can arrange for you to return
```

```
and will be forthcoming as soon as we receive your payment.
<FF>
If you are not completely satisfied with this product, please
notify us immediately so that we can arrange for you to return
```

Example 2: Sets the PAGE entity to be the string **.HL1**. Then moves the cursor to the next occurrence of the new page marker.

```
Now that you understand the basic concepts of word
processing, we can move on to an actual session at the terminal.
.HL1 Starting to Work with your Word Processing System
```

GOLD + PAGE COMMAND

```
Command:  SET ENTITY PAGE ".HL1"
```

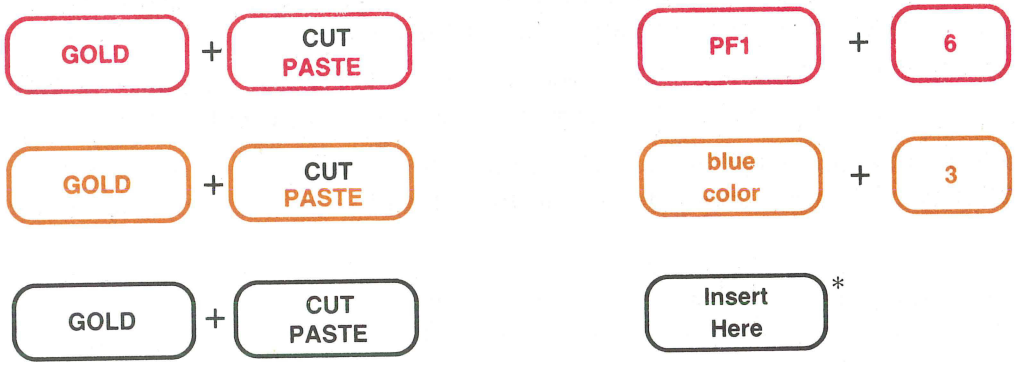
ENTER SUBS

PAGE COMMAND

```
Now that you understand the basic concepts of word
processing, we can move on to an actual session at the terminal.
.HL1 Starting to Work with your Word Processing System
```

Related Commands: Nokeypad – PAGE

Keypad Designations:



*On the LK201 keyboard, both the PF1/6 key sequence and the Insert Here key have the same preset function.

Description:

PASTE is used with **CUT** or **APPEND** to copy or move text within a buffer. PASTE copies the text currently residing in the PASTE buffer into the current buffer. The PASTE buffer contents is inserted to the left of the cursor regardless of EDT's current direction. PASTE has no effect on the contents of the PASTE buffer. **PASTE.** is the nokeypad definition for PASTE.

To move text from one place in your buffer to another, you need to use **SELECT**, CUT, and PASTE in the following order:

1. Use SELECT to create a select range of the text you want to move.
2. Press CUT to delete the text from the current buffer and put it in the PASTE buffer.
3. Move the cursor to the new location where you want to insert the deleted text.
4. Press PASTE to have EDT copy the PASTE buffer text into your current buffer to the left of the cursor.

You can use SELECT, CUT, and PASTE to copy text that exists in one place in your buffer to a second location. Follow the same procedure as for moving text, but add an additional step between the second and third steps:

2a. Press PASTE to have EDT restore the deleted text in its original location.

Each time you use CUT, EDT overwrites the contents of the PASTE buffer. If you want to add more text to the buffer before you insert it in the new location, you can use APPEND. APPEND deletes the select range text from its current location and adds it to the end of the PASTE buffer. When you press PASTE, both the text you deleted with CUT and the text you deleted with APPEND are inserted to the left of the cursor.

It is possible to edit the PASTE buffer. Using the line mode **FIND =buffer** command, you can enter the PASTE buffer, make your changes, and then return to your original buffer. Now when you use PASTE, the revised buffer contents are inserted at the cursor location.

When you use CUT to put part of a line into a buffer, EDT adds a line terminator at the end of the text since EDT buffers cannot hold partial lines. PASTE removes the added line terminator so that when you insert the text, you do not have an extra line terminator.

You can use the line mode FIND command to move from one buffer to another during your EDT session. Then you can use PASTE to put the contents of the PASTE buffer in that buffer.

Example 1: Uses a select range and CUT to move the second element of the list to the PASTE buffer. After moving the cursor, inserts the PASTE buffer contents to reorder the list.

```
Peripheral hardware, also known as input/output devices,
includes terminals, line printers, tapes, and disks.
```

SELECT RESET + WORD CHNGCASE + WORD CHNGCASE

```
Peripheral hardware, also known as input/output devices,
includes terminals, line printers, tapes, and disks.
```

CUT PASTE

```
Peripheral hardware, also known as input/output devices,
includes terminals, tapes, and disks.
```

(Move the cursor to the **a** of **and disks**.)

GOLD + CUT PASTE

```
Peripheral hardware, also known as input/output devices,
includes terminals, tapes, line printers, and disks.
```

Example 2: Uses a select range and CUT to move the separating line to the PASTE buffer. Immediately restores the deleted line to its former position. Moves the cursor to another location and copies the separating line there.

```
End of Section 1
***** ***** *****
```

SELECT RESET + LINE OPEN LINE + CUT PASTE + GOLD + CUT PASTE

```
End of Section 1
***** ***** *****
***** ***** *****
```

(Move the cursor to the next location where you want to put the line.)

```
End of Section 2
```

GOLD + CUT PASTE

```
End of Section 2
***** ***** *****
```

Example 3: Using a select range and CUT, moves three lines to the PASTE buffer. Then uses the line mode FIND command to enter the PASTE buffer and edit the text. Again, uses the line mode FIND command to return to the MAIN buffer. After finding the new location for the text, inserts the revised contents of the PASTE buffer.

```
You combine the control key with a character key by
pressing both simultaneously, i.e., hold the CTRL
key down while you press the character key.
```

SELECT RESET + LINE OPEN LINE + LINE OPEN LINE + LINE OPEN LINE + CUT PASTE

GOLD + PAGE COMMAND

```
Command:  FIND =PASTE
```

ENTER SUBS

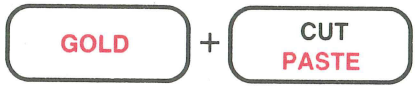
(Move cursor to the **i** of **i.e.** and delete the next four characters. Now type **that is** to replace **i.e.**.)

GOLD + PAGE COMMAND

```
Command:  FIND =MAIN
```

ENTER SUBS

(Move the cursor to the new location where you want to insert the text.)




```
You combine the control key with a character key by
pressing both simultaneously, that is, hold the CTRL
key down while you press the character key.
```

Related Commands: Line – COPY, MOVE
Nokeypad – PASTE

Syntax:

```
[count]PASTE[=buffer]
```

Description:

The PASTE command copies the contents of any buffer into the current buffer. The keypad **PASTE** function only copies the PASTE buffer contents into the current buffer.

When used together with the **CUT** or **APPEND** command, the nokeypad PASTE command enables you to move or copy text from one location to another in the same or different buffers. When you do not supply a buffer name with the PASTE command, EDT uses the PASTE buffer. Do not put a space between the word PASTE and the equal sign (=) when you are specifying another buffer name. Otherwise, EDT uses the PASTE buffer and tries to interpret the buffer name you specified as another command.

Note that the syntax for the PASTE command differs from the CUT command. You cannot use an entity specifier with PASTE; you can only copy entire buffers. However, remember that you can enter the PASTE buffer or another buffer and edit it before using PASTE to copy the contents into your current buffer. Use the line mode **FIND** command to move to the other buffer.

To move text, use the CUT command to delete the text from its present location. Then move the cursor to the new location and insert the text with the PASTE command. To copy text, use the CUT command to move the material you want to copy to the PASTE buffer. Then *before* moving the cursor, issue the PASTE command to restore the deleted text to its original location. Now move the cursor to the place where you want the copy inserted and issue the PASTE command again.

You can use the APPEND command to add text to the contents of the buffer you want to insert later with the PASTE command. When you move or copy text with CUT and PASTE or CUT, APPEND, and PASTE, be sure to use the same buffer name for all the commands. You can use the line mode **SHOW BUFFER** command to list all the buffers that currently exist in your editing session.

Each time you issue the CUT command, EDT overwrites the contents of the PASTE buffer or the specified buffer. When you use CUT to put part of a line into a buffer, EDT adds a line terminator at the end of the text since EDT buffers cannot contain partial lines. The PASTE command removes the added line terminator so that when you insert the text, you do not have an unwanted line break.

For information on how to modify the cursor position after EDT processes a PASTE command, see the nokeypad **KS** command.

Example 1: Inserts the text from the buffer PAGEBLANK at the current cursor location.

```
    in the following three chapters.
    <FF>
    □
```

```
PASTE=PAGEBLANK
```

```
    in the following three chapters.
    <FF>
    [This page intentionally left blank.]
    <FF>
    □
```

Example 2: After using CUT to delete the last line, inserts that line in the middle of the list.

```
    George G. Haraway
    John W. Hardwick
    Mason G. Harbaugh
```

```
CUTL
```

(Move the cursor to the **J** in **John**.)

```
PASTE
```

```
    George G. Haraway
    Mason G. Harbaugh
    John W. Hardwick
```

Example 3: Uses CUT, APPEND, and PASTE to alphabetize a list of cities.

```
    Dallas
    Houston
    Fort Worth
    Austin
    San Antonio
```

```
CUTL
```

(Move the cursor to the **D** in **Dallas**.)

```
APPENDL
```

(Move the cursor to the **F** in **Fort Worth**.)

```
APPENDL
```

(Move the cursor to the **H** in **Houston**.)

```
PASTE

    Austin
    Dallas
    Fort Worth
    Houston
    San Antonio
```

Related Commands: Keypad – PASTE
Line – COPY, MOVE

PRINT Command
Line

Syntax:

```
PRINT file-specification [=buffer] [range]
```

Description:

The PRINT command puts a copy of the specified range or buffer in an external file in the current or specified directory. EDT adds a form feed and two blank lines for every 60 lines it copies to the external file. Also, the EDT line numbers become part of the text in the external file. EDT does not resequence the lines when it processes the PRINT command. The line numbers remain exactly as they were before you issued the PRINT command.

When you specify neither range nor buffer, the entire current buffer is copied. If you give only a buffer name with no range, EDT copies the entire contents of the specified buffer, but does *not* move to that buffer. Instead, EDT remains in the buffer from which you issued the PRINT command.

Nonprinting characters are shown in the external file as EDT displays them, for example, <ESC> for the escape character (decimal value 27) and ^@ for the null character (decimal value 0).

If you include a directory name in the file specification, that directory must exist and you must have access to it. When you use the PRINT command, EDT creates a file, but EDT cannot create a directory.

The line numbers that appear in a file created by the PRINT command are different from those created by the **WRITE** command with the /SEQUENCE qualifier. See the section on EDT line numbers in Chapter 7 for more information.

Examples:

```
*PRINT MEMO1.RNO
```

Copies the entire current buffer to the file MEMO1.RNO.

```
*PRINT [JONES.STATUS]WEEK43 12 THRU END
```

Copies the current buffer starting with line 12 to the file WEEK43 in the subdirectory JONES.STATUS

```
*PRINT ADDRESS.DAT =HEADING
```

Copies the contents of the buffer HEADING to the file ADDRESS.DAT.

```
*PRINT LETTER1.RNO =BODY 7 THRU 56
```

Copies lines 7 through 56 from the buffer BODY to the file LETTER1.RNO.

Syntax:

```
/QUERY
```

Description:

The /QUERY qualifier gives you control over the processing of four commands: **COPY**, **DELETE**, **MOVE**, and **SUBSTITUTE**. Use the /QUERY qualifier when you are performing one of the four operations on a range of lines, but do not want every line in the range to be affected by the command. Always precede an EDT qualifier with a slash.

When you include the /QUERY qualifier in a command line, EDT prompts you for each line in the range before any action is taken. EDT prints the line and then displays the question mark prompt (?) so that you can decide whether to have EDT: (1) perform the command for that line, (2) skip the operation for that line, (3) process the command for all remaining lines in the range, or (4) stop processing the command at that line.

The four responses to the question mark prompt are:

Y	(YES)	Perform the command on this line.
N	(NO)	Do not perform the command on this line.
A	(ALL)	Perform the command on all remaining lines in the range.
Q	(QUIT)	Stop performing the command at this line. The remaining lines in the range will not be affected.

When you use the /QUERY qualifier with the SUBSTITUTE command, it might seem that there are four delimiters on the command line.

```
SUBSTITUTE/1983/1984/ WHOLE /QUERY
```

The slash before the word **QUERY** is the qualifier signal, not a delimiter. If you use a different delimiter for the substitute strings, you can see the distinction:

```
SUBSTITUTE*1983*1984* WHOLE /QUERY
```

Example: Prompts you to find out which lines in the range 5 through 9 you want to delete. The TYPE command shows which lines in the range 4 through 10 remain in the buffer.

```
     4        Jacksonville, FL
     5        Huntsville, AL
     6        Thomasville, GA
     7        Potsdam, NY
     8        Grand Forks, ND
     9        Missoula, MT
    10        Eugene, OR

*DELETE 5 THRU 9 /QUERY

     5        Huntsville, AL
?Y
     6        Thomasville, GA
?N
     7        Potsdam, NY
?Y
     8        Grand Forks, ND
?A
4 lines deleted
    10        Eugene, OR

*TYPE 4 THRU 10

     4        Jacksonville, FL
     6        Thomasville, GA
    10        Eugene, OR
```

Syntax:

```
QUIT [/SAVE]
```

Description:

The QUIT command ends an EDT session without copying any text to an external file. Only a copy of the original file as it was before you started your EDT session exists. There is no copy of the edited text. If you use EDT to create a new file and you type QUIT to end your editing session, no copy of the text will exist in any directory. Use the **EXIT** command to preserve a copy of the MAIN buffer text in an external file.

If you use the /**SAVE** qualifier with QUIT, EDT saves the journal file from your editing session. You might want to do this if your text has become confused or if you accidentally deleted some text that you want to recover. First use QUIT/SAVE to end your editing session. Then using EDT, delete the last few commands from the journal file. To resume your work, use the /RECOVER qualifier with your EDIT/EDT command to restore your editing work up to the point where the problem occurred. For information on how to edit the journal file, see Chapter 7.

Pressing CTRL/Z three times in a row has the same effect as issuing the QUIT/SAVE command.

Example 1: Ends your EDT session without copying the MAIN buffer contents to an external file. Only the system prompt appears.

```
*QUIT

☞
```

Example 2: Lists the files in the directory. Then starts an EDT session, but ends it with QUIT/SAVE to retain the journal file. A second system DIRECTORY command shows the journal file (CHAPTER1.JOU) in the directory.

```
☞ DIRECTORY

    CHAPTER1.RNO;2  CHAPTER1.RNO;1

☞ EDIT /EDT CHAPTER1.RNO
  .
  .
  .
*QUIT/SAVE

☞ DIRECTORY

    CHAPTER1.JOU;1  CHAPTER1.RNO;2  CHAPTER1.RNO;1
```

Related Commands: Nokeypad – QUIT

QUIT Command
Nokeypad

Syntax:

```
QUIT
```

Description:

The QUIT command ends your nokeypad editing session without saving your edits and without passing through line mode. You are no longer in EDT, no new file is saved in any directory, and your terminal is now at the system command level.

The nokeypad QUIT command cannot take the /**SAVE** qualifier that the line mode **QUIT** command has. The nokeypad QUIT command can only end your editing session.

Example: Ends the nokeypad editing session without copying the MAIN buffer contents to an external file. Only the system prompt appears.

```
QUIT

    ☞
```

Related Commands: Line – QUIT

Syntax:

```
[+|-][count]R[+|-][count]entity
```

Description:

The R (replace) command combines the delete and insert functions. First EDT deletes the specified text; then it shifts to the insert state so that you can enter text at your terminal. When you finish entering the new text, press **CTRL/Z** to exit from the insert state. While you are inserting the new text, EDT continues to display the R command at the bottom of the screen. The entity specifier always refers to the text you want to delete.

Example: Deletes the last line and replaces it with new text.

```
This seminar will cover the following topics:

     Management by Objective
     Management by Design
     Management by Intimidation

RL
```

(Now type the new line, **Management by Consensus**, and complete the insert by pressing CTRL/Z.)

```
This seminar will cover the following topics:

     Management by Objective
     Management by Design
     Management by Consensus
```

Related Commands: Keypad – REPLACE
Line – REPLACE

range specifier
Line

Syntax:

```
range-indicator
```

Description:

The range specifier is used with these line mode commands:

CHANGE	<null>
COPY	PRINT
DELETE	REPLACE
FIND	RESEQUENCE
INCLUDE	SUBSTITUTE
INSERT	TYPE
MOVE	WRITE

The range specifier can reference one line or a group of lines depending on your needs or the command with which you use the range.

There are a number of ways to indicate a range. Some of the more common ones are listed here. See Table 4-1 for a complete list. When you use the string specifier with most line mode commands, EDT considers that you are referencing the entire line. For example, if you issue the command **DELETE "string"**, EDT deletes the entire line that contains the quoted string.

EDT keeps track of the string location when you use a string specifier with a command like **FIND** or **TYPE**. By remembering the position of the string, EDT enables you to locate the next occurrence of that string.

Range	Description
. (period)	The current line
number	The exact line number, for example, 95.31
range THRU range range:range	The lines starting at the first range indicator and continuing through the second range indicator
WHOLE	Every line in the buffer
BEGIN	The top of the buffer
END	The bottom of the buffer
BEFORE	The first line in the current buffer through the line above the current line
REST	The current line in the current buffer through the last line in the buffer
SELECT	The active select range established by the keypad **SELECT** function or the nokeypad **SEL, SSEL,** or **TGSEL** command. The select range can only include whole lines.

(Continued on next page)

Range	Description
"string" 'string'	A string of characters located on or after the current line
–"string" –'string'	A string located before the current line
+n	The number of lines below the current line; for example, +3 is the third line below the current one
–n	The number of lines above the current line; for example, –10 is the tenth line above the current one

When a string is used for the range specifier, it must be surrounded by either single (') or double (") quotation marks. When the first element of the <null> command is a word range specifier, such as BEGIN, END, BEFORE, REST, or WHOLE it must be preceded with a percent sign (%) to indicate to EDT that it is not a command or EDT macro.

Examples:

```
*
```
Types the current line.

```
*CHANGE "page"
```
Shifts to a change mode and puts the cursor to the right of the next occurrence of the word **page**.

```
*DELETE 25 THRU END
```
Deletes lines starting with 25 through the end of the buffer.

```
*FIND =PASTE 23
```
Moves to line 23 in the PASTE buffer.

```
*%REST
```
Displays the remaining lines in the current buffer, starting with the current line.

```
*WRITE OUTFILE.DAT . THRU +10
```
Creates a file called OUTFILE.DAT and copies 11 lines into it, starting with the current line and ending with the tenth line after the current line.

```
*RESEQUENCE 25 -5 THRU 30 /SEQUENCE:100
```
Resequences the lines starting with five lines before line number 25 and continuing through line number 30 or as far as is necessary to maintain ascending line numbers. Assigns the number 100 to the first line in the range. Then increments the line numbers by 1.

```
*WRITE LETTER1A.RNO "In response" THRU "cc:  David North"
```
Creates a file called LETTER1A.RNO and puts in it a copy of the lines starting with the one containing the string **In response** through the one with the string **cc: David North**.

REF (Refresh) Command
Nokeypad

Syntax:

```
REF
```

Description:

The REF command clears and redraws the entire screen. Use REF to eliminate any extraneous characters or messages that have appeared on the screen but are not part of the current text you are editing. After the REF command is given, EDT discards any characters that are not part of the current text. The screen is restored to its state prior to the interruption. REF has no effect on the text you are editing.

Example: Refreshes the screen to eliminate the notification of new mail.

```
and expect to hear from you in the near future.
>NEW MAIL FROM YYYYYY::WHITEif you have any questions or
problems.

REF

and expect to hear from you in the near future.
In the meantime, please call if you have any questions or
problems.
```

Related Commands: Keypad – CTRL/R, CTRL/W

Keypad Designations:

Description:

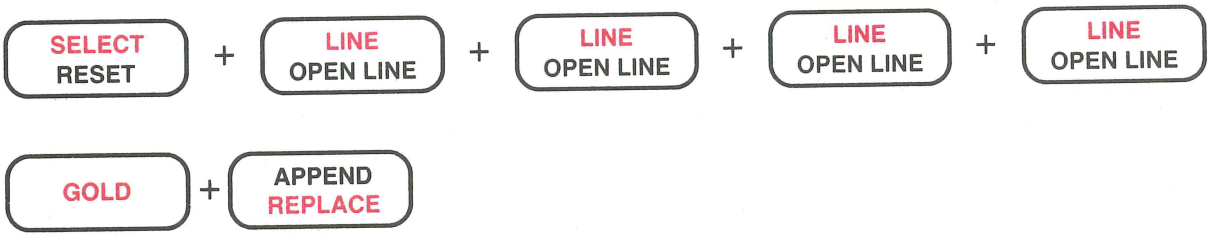
REPLACE deletes the text in the select range and replaces it with the contents of the PASTE buffer. REPLACE enables you to delete different blocks of text, but replace them all with the same text. EDT stores the deleted text in a buffer called DELETE. If the buffer does not exist, EDT creates it. If you have created a buffer called DELETE, EDT overwrites the text you had in that buffer with the newly deleted text. Each time you use REPLACE, EDT always overwrites the text in the DELETE buffer. The DELETE buffer can be entered and edited and its name appears on the **SHOW BUFFER** list.

You can use **CUT** to put the replacement text into the PASTE buffer, or you can move to the PASTE buffer with the line mode **FIND** command and insert the text directly there.

CUTSR = DELETE PASTE. is the nokeypad definition for REPLACE.

Example: Creates a select range of four lines. Moves the selected text to the DELETE buffer and inserts the contents of the PASTE buffer.

```
Martha Jackson
Purchasing Department
Kitchen Cabinets, Inc.
Post Mills, VT
```



REPLACE (Cont.)
Keypad

```
Judy Henning
Purchasing Department
Kitchens Incorporated
Cumberland, RI  02864
```

Related Commands: Line – REPLACE
Nokeypad – R (replace)

Syntax:

```
REPLACE [=buffer] [range] [;line-to-be-inserted]
REPLACE [=buffer] [range] RETURN text CTRL/Z
```

Description:

The REPLACE command deletes the line(s) specified by range from the current or specified buffer. EDT then replaces the deleted lines with text that you enter at the terminal. There are two ways to enter new text: (1) on the same line as the REPLACE command, or (2) through the insert state.

If the text to be entered does not contain any line terminators, it can be typed on the same line as the REPLACE command. Separate the text from the command with a semicolon (;). When you press RETURN, the line or lines specified by range are deleted and replaced by the text following the semicolon.

To replace the deleted text with one or more lines, press RETURN after typing the REPLACE command. EDT shifts to the insert state as soon as it deletes the specified lines. Use **CTRL/Z** to exit from the insert state when you have finished entering the new text.

Only the single line form of the REPLACE command can be used in startup command files or EDT macros.

Example 1: Using the single line form of the REPLACE command, deletes line 3 and replaces it with new text. The TYPE command verifies that the new line has been entered.

```
1       Mr. Theodore R. Swenson
2       34 North Main Street
3       Londonderry, NH  03053
4       (603) 555-1234

*REPLACE "London" ;Derry, NH  03038
1 line deleted
4       (603) 555-1234

*TYPE -1
3       Derry, NH  03038
```

Example 2: Deletes line 7 and inserts new text. The TYPE command verifies the new text.

```
    6       Robert Harrison
    7       John Barber

*REPLACE 7 (RETURN)
1 line deleted

            Marjorie Dickerson
            James Newbold
            Dorothy Urquart
            Faith Jaspersen
            (CTRL/Z)

    8       William Wetherall

*TYPE 6 THRU 8

    6       Robert Harrison
    6.1     Marjorie Dickerson
    6.2     James Newbold
    6.3     Dorothy Urquart
    6.4     Faith Jaspersen
    8       William Wetherall
```

Example 3: Using the single line form of the REPLACE command, moves to the buffer HEADING and replaces the first line with new text. Verifies the new text with the TYPE command.

(This is the contents of the buffer HEADING.)

```
    1       DATE:   July 8, 1985
    2       TO:     Sam Hill
    3       FROM:   John Archer
```

(The REPLACE command is typed from the MAIN buffer.)

```
*REPLACE =HEADING 1;DATE:  August 20, 1985
1 line deleted

    2       TO:     Sam Hill

*TYPE 1
    1       DATE:   August 20, 1985
```

Related Commands: Keypad – REPLACE
Nokeypad – R (replace)

Syntax:

```
RESEQUENCE [=buffer] [range] [/SEQUENCE[:initial[:increment] ] ]
```

Description:

The RESEQUENCE command assigns new EDT line numbers to the lines of the current buffer or the specified buffer. If you use the buffer specifier without a range specifier, EDT renumbers the entire buffer. If a range is given, the lines within that range must be contiguous.

When RESEQUENCE is used without the /**SEQUENCE** qualifier, EDT uses the value 1 for both of the /SEQUENCE specifiers – :initial and :increment. You get the same results if you use RESEQUENCE /SEQUENCE with no :initial and :increment specifiers. You can only omit the /SEQUENCE qualifier and its specifiers when you are renumbering an entire buffer or when the first line specified by **range** has a number less than or equal to 1.

After EDT renumbers your text, it prints the number of lines that have been resequenced. This number reflects the actual number of lines that EDT had to resequence in order to avoid duplicating or overlapping line numbers.

Examples:

```
*RESEQUENCE
*RESEQUENCE /SEQUENCE
*RESEQUENCE /SEQUENCE:1:1
```

All three commands resequence the entire current buffer using default increment and initial values of :1:1.

```
*RESEQUENCE =BUFF3
```

Resequences all of buffer BUFF3 using the default values :1:1.

```
*RESEQUENCE /SEQUENCE:100:10
```

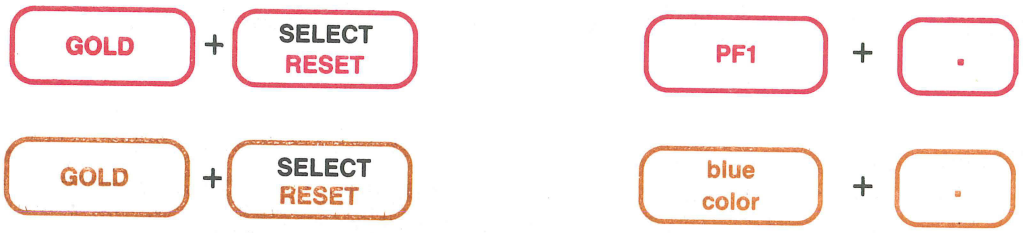
Resequences the entire current buffer starting with an initial line number of 100 and using increments of 10.

```
*RESEQUENCE =LIST 5 THRU END /SEQUENCE:10:10
```

Moves to the buffer LIST and resequences it starting at line 5. Assigns the initial value of 10 to line 5 and then increments the successive line numbers by 10.

RESET Function
Keypad

Keypad Designations:



Description:

RESET changes several conditions of your editing session:

- Cancels an active select range.
- Sets EDT's current direction to **ADVANCE**.
- Sets EDT to the default **DMOV** state.

RESET also has a special function within the **CTRL/K** key definition facility. Namely, you can use RESET to delete the text on the definition line if you want to start your definition over again.

RESET is the nokeypad definition for RESET. Note that there is no period at the end of the definition. This is because RESET is not a nokeypad command.

Example 1: Cancels the active select range.

```
The next meeting is scheduled for December 13th in my office.
```

```
The next meeting is scheduled for December 13th in my office.
```

```
The next meeting is scheduled for December 13th in my office.
```

Example 2: Uses RESET to delete to beginning of line so that you can retype the key definition correctly. Shows how to create a key definition to access nokeypad commands directly from keypad mode.

CTRL/K

```
Press the key you wish to define
```

CTRL + N

```
Now enter the definition terminated by ENTER
?'Bijetoad cinn
```

GOLD + SELECT RESET

```
Now enter the definition terminated by ENTER
?'Nokeypad command: '.
```

Related Commands: Nokeypad – DESEL

RETURN Function
Keypad

Keypad Designations:

Description:

RETURN adds a line terminator to the text you are editing. The new line terminator is inserted to the left of the current cursor position. The cursor remains on the same character where it was before you pressed RETURN. If the cursor is at the beginning of the line, a blank line is created above the current cursor line.

When the cursor is in the middle of a line, RETURN moves the cursor character and all the text to the right of the cursor to a new line. When the cursor is at the end of a line, RETURN adds a line terminator, creating a blank line below the current line. The cursor is now positioned at the beginning of the new blank line. **OPEN LINE** also inserts a line terminator in your text, but it positions the cursor to the new line terminator.

You can redefine the RETURN key, although this is not recommended. When you redefine the RETURN key, you also redefine **CTRL/M**. To find out the definition of the RETURN key, type **SHOW KEY** CONTROL M.

Example 1: Inserts a blank line between the first and second lines.

```
Meeting Report
On Wednesday, I attended a seminar on the advanced
capabilities of the EDT Editor.  The topics covered
included startup command files, defining and
redefining keys, defining and using macros.
The presentation was very good.
```

```
Meeting Report

On Wednesday, I attended a seminar on the advanced
capabilities of the EDT Editor.  The topics covered
included startup command files, defining and
redefining keys, defining and using macros.
The presentation was very good.
```

Example 2: Using the same text, inserts a blank line above the last line.

(Move the cursor to the beginning of the last line.)

```
Meeting Report

On Wednesday, I attended a seminar on the advanced
capabilities of the EDT Editor.  The topics covered
included startup command files, defining and
redefining keys, defining and using macros.

The presentation was very good.
```

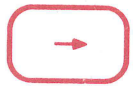
Example 3: Using the same text, inserts a line terminator that moves the words **The topics covered** to a new line.

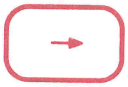
(Move the cursor to the **T** in **The topics covered**.)

```
Meeting Report

On Wednesday, I attended a seminar on the advanced
capabilities of the EDT Editor.
The topics covered
included startup command files, defining and
redefining keys, defining and using macros.

The presentation was very good.
```

Right Arrow
Keypad
Nokeypad

Keypad Designations:





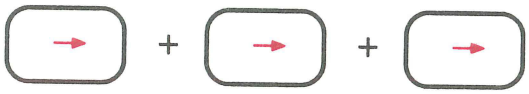
Description:

Right Arrow moves the cursor one character to the right, regardless of EDT's direction. **+C.** is the nokeypad definition for Right Arrow.

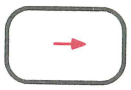
If the cursor is on a line terminator, Right Arrow moves the cursor to the first character on the next line.

Example: Moves the cursor to the right to the end of first line. Then moves the cursor to the beginning of the second line.

```
January 1, 1985
April 1, 1985
```



```
January 1, 1985
April 1, 1985
```



```
January 1, 1985
April 1, 1985
```

Syntax:

```
[+|-][count]S/[string-1]/string-2/
```

Description:

The S (substitute) command looks for the next occurrence of string-1, deletes it, and replaces it with string-2.

The count specifier instructs EDT to perform the specified number of searches and substitutions. If the count exceeds the total number of occurrences of string-1 between the current cursor position and either the beginning or end of the buffer (depending on the direction specified), EDT performs as many substitutions as it can, and then prints the message **String was not found** to indicate that it could not find another string-1. The cursor is now located just after the last substitution that was made.

The sign specifier enables you to change the direction of the search for string-1, for that S command only, without affecting EDT's direction.

The slashes are string delimiters. Any nonalphanumeric character can be used for a string delimiter as long as that character is not contained in either string-1 or string-2. All three delimiters must be identical.

String-1 or string-2 can sometimes be omitted from the S command. If you omit string-1, EDT always uses the current search string. If the search buffer is empty, EDT simply inserts string-2 into the text to the left of the cursor. If you omit string-2, EDT always deletes string-1 and inserts nothing. The following samples show what happens when you omit string-1, string-2, or both:

```
This is file A.      S/file/buffer/      This is buffer A.
This is file B.      S///                This is  B.
This is file C.      S/file//            This is  C.
This is file D.      S//buffer/          This is buffer D.
This is file E.      S                   This is  E.
```

The **SN** command, which does not take any strings as specifiers, always uses the current contents of the search buffer and the substitute buffer.

EDT performs searches in several ways. For more information see the **SET SEARCH** command in this chapter and the section on SET SEARCH in Chapter 6. **SHOW SEARCH** tells which search parameters are currently in effect.

Example 1: Changes **1987** to **1988**.

```
January 3, 1987
S/7/8/
January 3, 1988
```

Example 2: Changes **pages** to **Sections** in two places.

```
    In pages 17 through 25.  Then in pages 32 through 40 we

2S/pages/Sections/

    in Sections 17 through 25.  Then in Sections 32 through 40 we
```

Example 3: With SET SEARCH EXACT, moves back to the first line to change **users** to **you**, skipping over **Users** on line 2.

```
    If users are able to access the line printer, then the DCL
    PRINT command can be given.  Users who only have access to
    their terminals are restricted to the DCL TYPE command.

EXT SET SEARCH EXACT

-S/users/you/

    If you are able to access the line printer, then the DCL
    PRINT command can be used.  Users who only have access to
    their terminals are restricted to the DCL TYPE command.
```

Related Commands: Keypad – SUBS
Line – SUBSTITUTE

Syntax:

```
/SAVE
```

Description:

The /SAVE qualifier instructs EDT to save a copy of the journal file in the current directory. The journal file contains a record of all the keystrokes you have made during the current EDT session.

/SAVE is used with the line mode commands **EXIT** and **QUIT**. When a normal exit is made from EDT, whether by means of EXIT or QUIT, EDT discards the journal file. The /SAVE qualifier causes the journal file to be saved in your current directory. The journal file is automatically saved whenever your editing session ends abnormally.

All EDT qualifiers must be preceded by slashes.

Examples:

```
*EXIT /SAVE
```

Exits from EDT saving a copy of the MAIN buffer as well as the journal file.

```
☞ EDIT /EDT LIST.DAT
    .
    .
    .
*EXIT NEWLIST.DAT /SAVE
```

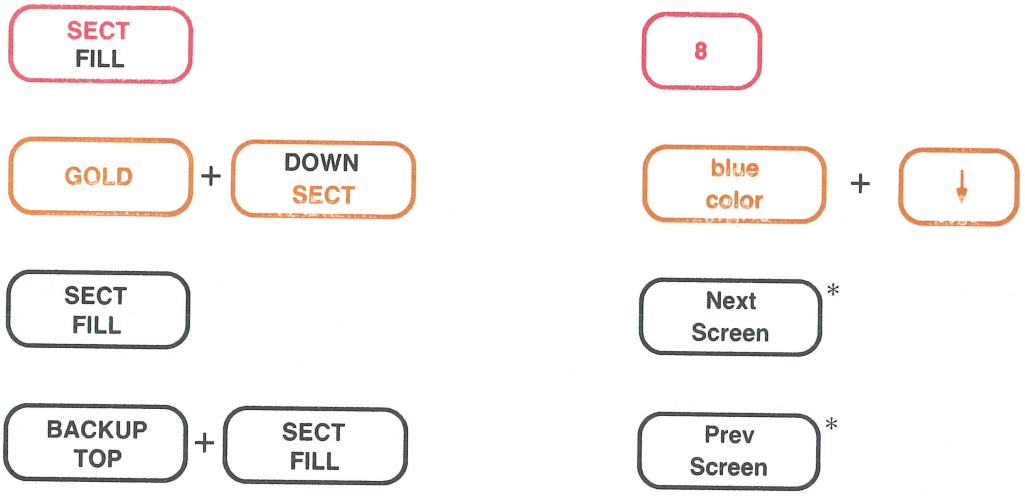
Exits from EDT saving a copy of the MAIN buffer in a file called **NEWLIST.DAT**. The journal file is called **LIST.JOU**.

```
*QUIT /SAVE
```

Exits from EDT without saving a copy of the MAIN buffer. Only the journal file is saved.

SECT (Section) Function
Keypad

Keypad Designations:



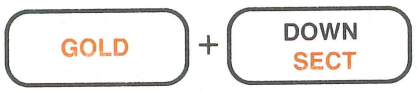
*On the LK201 keyboard, the **Next Screen** key moves the cursor 16 lines forward, regardless of EDT's current direction. The **Prev Screen** key moves the cursor 16 lines backward, regardless of EDT's current direction. On all screen mode terminals the SECT function moves the cursor 16 lines in the current direction.

Description:

SECT (section) moves the cursor one section – 16 lines – toward the end or beginning of the buffer, depending on EDT's current direction. The cursor is always placed at the beginning of the new current line regardless of its previous position. **(16L)**. is the nokeypad definition for SECT.

Example: Moves the cursor from its position at the end of Test Line 1 to the beginning of Test Line 17.

```
Test Line  1
Test Line  2
Test Line  3
Test Line  4
Test Line  5
Test Line  6
Test Line  7
Test Line  8
Test Line  9
Test Line 10
Test Line 11
Test Line 12
Test Line 13
Test Line 14
Test Line 15
Test Line 16
Test Line 17
```




```
Test Line  1
Test Line  2
Test Line  3
Test Line  4
Test Line  5
Test Line  6
Test Line  7
Test Line  8
Test Line  9
Test Line 10
Test Line 11
Test Line 12
Test Line 13
Test Line 14
Test Line 15
Test Line 16
Test Line 17
```

SEL (Select) Command
Nokeypad

Syntax:

```
SEL
```

Description:

The SEL (select) command marks the current cursor position as one end of a select range. To set up the select range, first position the cursor at one end of the text you want to select. Then issue the SEL command to have EDT mark that spot. Use the arrow keys or the "**move**" command to move the cursor to the other end of the selected text. The select range is now set. To use that select range, use SR (select range) as the entity specifier with your nokeypad mode command.

You can move the cursor either forward or backward in the buffer to set the other end of the select range once the initial end has been marked. To move backward toward the start of the buffer, use the Left and Up arrow keys or precede the "move" command with a minus sign (–). If you have included more text than you want in the select range, you can move the cursor toward the position marked by SEL using the arrow keys and "move" commands to reduce the size of the range. Adjusting select ranges on VT100 terminals is easy because EDT displays the text in reverse video. On VT52 terminals, you might find it easier to use the DESEL command to cancel the select range and then start over.

The **DESEL** (deactivate select) command cancels a select range. The **TGSEL** (toggle select) command allows you to switch between SEL and DESEL with the same command, either setting SEL if no select range is active, or DESEL if there is a current select range. The **SSEL** (search and select) command performs a search and makes the found string the select range.

You can use a select range with line mode commands by giving the line mode range specifier SELECT. However, line mode requires that the select range include only whole lines.

Example 1: Creates a select range of the word **Digital** and then changes all lowercase letters to uppercase.

```
     This manual was produced by Digital for its customers.
SEL W
CHGUSR
     This manual was produced by DIGITAL for its customers.
```

Example 2: Creates a select range starting near the end of the third line and including the first two lines. Deletes the unwanted text.

```
The reason for this change of plans should be obvious:  no one
in his right mind would want to have a meeting in North Alaska in
the middle of winter even if he were an avid skier.  The meeting
is now relocated to Lake Tahoe.  For those of you wishing to see
the 49th state, our summer meeting is scheduled for Fairbanks.
```

```
SEL

BL -2L

DSR
```

```
The meeting
is now relocated to Lake Tahoe.  For those of you wishing to see
the 49th state, our summer meeting is scheduled for Fairbanks.
```

Related Commands: Keypad – SELECT

SELECT Function
Keypad

Keypad Designations:

*On the LK201 keyboard, both the period (.) keypad key and the **Select** key have the same preset function.

Description:

SELECT sets up a select range for use with keypad functions such as **APPEND**, **CHNGCASE**, **CUT**, **FILL**, **REPLACE**, **SUBS**, and **CTRL/T**. Start with the cursor at one end of the text you want selected. Next press SELECT to mark that position as the beginning of the select range. Then, using the arrow keys and/or function keys that move the cursor, mark the other end of the text being selected. Now you are ready to press a function key that uses a select range.

The **RESET** function cancels the select range. If you have included more text than you wanted in the select range, you can move the cursor back toward the position initially marked by SELECT, using arrow keys and cursor moving functions, thus reducing the size of the range. Adjusting select ranges on VT100 terminals is easy because EDT shows the text in reverse video. On VT52 terminals, you might find it easier to use RESET to cancel the select range and then start over.

You can use a select range with line mode commands by giving the line mode range specifier SELECT. However, line mode requires that the select range contain only whole lines.

SEL. is the nokeypad definition for SELECT.

Example 1: Creates a select range of the last line and, using CUT, moves it to the PASTE buffer. Uses PASTE to insert the line in its proper alphabetical order.

```
Introduction to BASIC
Roget's Thesaurus
American Heritage Dictionary
```

```
Introduction to BASIC
Roget's Thesaurus

```

```
American Heritage Dictionary
Introduction to BASIC
Roget's Thesaurus
```

Example 2: First uses the COMMAND function to process the line mode SET WRAP 60. Then creates a select range of four lines. Reformats the select range text with the FILL function.

```
In order to process data, there must be
communication between you and the computer.  This communication
is achieved by means of the
two fundamental computer components:  hardware and software.
```

```
Command:  SET WRAP 60
```

SELECT (Cont.)
Keypad

GOLD + PAGE COMMAND

```
Command:  SET WRAP 60
```

ENTER SUBS

SELECT RESET + LINE OPEN LINE + LINE OPEN LINE + LINE OPEN LINE + LINE OPEN LINE

CTRL/F

```
In order to process data, there must be communication
between you and the computer.  This communication is
achieved by means of the two fundamental computer
components:  hardware and software.
```

Related Commands: Nokeypad – SEL (select)

Syntax:

```
/SEQUENCE[:initial[:increment]]
```

Description:

The /SEQUENCE qualifier is used with the line mode commands **EXIT**, **RESEQUENCE**, and **WRITE**. Using the /SEQUENCE qualifier enables you to renumber the EDT line numbers for a block of lines. With the EXIT and WRITE commands, the /SEQUENCE qualifier causes sequence numbers to be written to the external file created by those commands.

With the RESEQUENCE command, the :initial specifier is the line number you want assigned to the first line of the specified buffer or range. The default value for :initial is 1. The :increment specifier refers to the spacing you want between the numbers in the new sequence. The default value for :increment is also 1. You must include the :initial specifier if you want to use :increment.

When you use the /SEQUENCE qualifier with EXIT or WRITE, the default values for :initial and :increment are different. The default values are the current EDT line numbers for the block of lines being copied to the external file. However, line numbers with decimal fractions cannot be stored in a sequenced file, so EDT must adjust any decimal line numbers to be whole numbers when you use the /SEQUENCE qualifier with no specifiers. For details on the function of the /SEQUENCE qualifier with the EXIT and WRITE commands, see the section on EDT line numbers in Chapter 7.

All EDT qualifiers must be preceded by slashes.

Examples:

```
*RESEQUENCE
*RESEQUENCE /SEQUENCE
*RESEQUENCE /SEQUENCE:1:1
```

All of these RESEQUENCE commands renumber the entire current buffer starting with 1 and incrementing by 1.

```
*EXIT /SEQUENCE:100:10
```

Exits from EDT. The output file has sequence numbers that start at 100 and increase by 10.

```
*WRITE LIST.DAT 25 THRU END /SEQUENCE:10:5
```

Writes a copy of lines 25 through the end of your current buffer to an external file called LIST.DAT. The file has sequence numbers starting at 10 for line 25 in the current buffer and incrementing by 5.

SET [NO]AUTOREPEAT Command
Line

Syntax:

```
SET AUTOREPEAT
SET NOAUTOREPEAT
```

Description:

The SET AUTOREPEAT command enables EDT to use the DECARM VT100 control sequence to prevent keypad keys (including arrow keys) from repeating faster than EDT can update the screen. SET AUTOREPEAT is the default.

On some VT100-type terminals, SET AUTOREPEAT causes the arrow keys to repeat very slowly (approximately .5 seconds for each repeated use). With SET NOAUTOREPEAT, the arrow keys repeat faster because EDT does not use the DECARM control sequence. If EDT gets behind, it simply skips intermediate screen updates.

Occasionally, on the VT100 terminal with printer port (but not on the VT102, which has an integrated printer port), use of SET NOAUTOREPEAT causes the terminal to stop transmission. If this occurs, press the SET-UP key (located at the upper lefthand corner of the keyboard) twice to clear the keyboard buffer.

SET [NO]AUTOREPEAT has no effect on the keypad mode GOLD/repeat feature or on the SPECINS function. These operations are affected by **SET [NO]REPEAT**.

Example: Changes the default SET AUTOREPEAT setting to SET NOAUTOREPEAT.

```
*SET NOAUTOREPEAT
```

Syntax:

```
SET CASE NONE
SET CASE LOWER
SET CASE UPPER
```

Description:

The SET CASE command is useful when you are reading text which has both upper- and lowercase letters at a single-case terminal. The SET CASE command uses flags to distinguish lowercase letters from uppercase ones. If you use SET CASE UPPER, each uppercase letter in the original file is preceded by an apostrophe ('). When you specify SET CASE LOWER, each lowercase letter in the original file is marked by an apostrophe. The flags appear only when the text is displayed at a terminal under line mode. No permanent marks are placed in the text itself.

The default state for EDT is SET CASE NONE, meaning that no provisions are made for artificially distinguishing between upper- and lowercase letters.

Example: Assumes that you are working at a single-case terminal. Issues the SET CASE UPPER command to flag all letters that are actually uppercase in the original text.

```
  USE THE SET CASE UPPER COMMAND WITH A SINGLE-CASE TERMINAL.

*SET CASE UPPER

  'USE THE 'S'E'T 'C'A'S'E 'U'P'P'E'R COMMAND WITH A SINGLE-CASE TERMINAL.
```

SET COMMAND Command Line

Syntax:

```
SET COMMAND file-specification
```

Description:

The SET COMMAND command allows you to process additional startup command files at the beginning of your EDT session. The word COMMAND refers to the startup command file, as does the /COMMAND qualifier used in EDIT/EDT command lines. You can only use the SET COMMAND command in a startup command file.

The default startup command file that EDT looks for is a system-wide file located in the system library. The file specifications are:

SYS$LIBRARY:EDTSYS.EDT	VAX/VMS
LB:EDTSYS.EDT	RSTS/E
LB:[1,2]EDTSYS.EDT	RSX-11M/M-PLUS

When you issue the system EDIT/EDT command, EDT first checks the command line to see if you have included a command file specification. If EDT finds one, it processes that file. Otherwise, EDT looks for a system-wide startup-command file to process. If no system-wide command file exists, EDT looks for a file named EDTINI.EDT in your default directory and processes that file. If you use the /NOCOMMAND qualifier in your EDIT/EDT command line, or if EDT fails to find a startup command file, your session begins with the default settings for all parameters.

You must include at least the file name in the SET COMMAND command line. **.EDT** is the default file type. If the specified file does not exist, EDT ignores the SET COMMAND command and continues to process any remaining commands in the current startup command file. Thus, you can have several SET COMMAND commands in the system startup command file, each branching to a different group or personal startup command file. For more information on startup command files and the SET COMMAND command, see Chapter 7.

To bypass the default startup command file, either specify a different startup command file in the EDIT/EDT command line or use the /NOCOMMAND qualifier in the EDIT/EDT command line. More information about designating startup command files on the EDIT/EDT command line appears in the system appendixes.

If you include the SHOW COMMAND command in a startup command file, the name of that file is displayed at your terminal when EDT reads the file.

Example: Shows the SET COMMAND entry in a system-wide startup command file, branching to the personal command file EDTEXPERT.EDT. Another SET COMMAND command at the end of the file transfers EDT to the EDTINI.EDT command file for users who do not have an EDTEXPERT.EDT command file in their default directories.

```
DEFINE KEY GOLD S AS "S^@?'Find: '^@?' Substitute: '^@."
SET WRAP 65
  .
  .
  .
SET COMMAND EDTEXPERT.EDT
DEFINE KEY GOLD L AS "SHL."
DEFINE KEY GOLD R AS "SHR."
  .
  .
  .
SET COMMAND EDTINI.EDT
```

SET CURSOR Command
Line

Syntax:

```
SET CURSOR top:bottom
```

Description:

The SET CURSOR command controls the scrolling of the screen relative to the cursor position. SET CURSOR has no effect if you are using a screen terminal in line mode.

The **top** specifier refers to the number of lines from the top of the screen to the cursor. As you move the cursor up toward the beginning of the buffer, EDT begins to scroll the screen display. Lines are added at the top of the screen when the cursor is the specified number of lines from the top. The **bottom** specifier refers to the number of lines from the cursor to the bottom of the screen. As you move the cursor down toward the end of the buffer, EDT begins to scroll the screen display. Lines are added at the end of the screen when the cursor is the specified number of lines from the bottom.

The default cursor settings are 7:14. When you move the cursor down to the 15th line on the screen, the first line on the screen disappears and the next line after the current bottom line appears at the bottom of the screen. Conversely, as you move the cursor up the screen, when it reaches the seventh line from the top, EDT begins scrolling down. Of course, scrolling stops when you are near the beginning or end of the buffer.

When using the SET CURSOR command, **top** must be a smaller number than **bottom**, and both must be less than or equal to the total number of screen lines.

The maximum number of lines on a screen is 22, designated 0 through 21 for the SET CURSOR command. The minimum value for the top specifier is 0 and the maximum number for the bottom specifier is 21.

You can use the **SET LINES** command to reduce the maximum number of lines visible on your screen.

Examples:

```
*SET CURSOR 1:20
```

The cursor remains either one line from the top of the screen or from the bottom of the screen, depending on the scrolling direction.

```
*SET CURSOR 9:11
```

The cursor remains one line on either side of the middle of the screen, depending on the scrolling direction.

Syntax:

```
SET ENTITY WORD "string"
SET ENTITY SENTENCE "string"
SET ENTITY PARAGRAPH "string"
SET ENTITY PAGE "string"
```

Description:

The SET ENTITY command defines the delimiters that mark certain entity boundaries for EDT commands and functions. You can use the SET ENTITY command to redefine the boundaries for four screen mode entities: WORD, SENTENCE, PARAGRAPH, AND PAGE. Other entities, such as character and line, cannot be redefined.

The default delimiters for the four EDT entities that can be reset are:

```
WORD                    <LF><VT><FF><CR>
SENTENCE          .?!
PARAGRAPH         <CR><CR>
PAGE              <FF>
```

A WORD is defined as any group of characters bounded by a space, horizontal tab (displayed by EDT as if it were a group of spaces), a line feed, a vertical tab, a form feed, or a carriage return. A SENTENCE is defined as any group of characters bounded by a period, a question mark, or an exclamation point, each of which must be followed by at least one space or line terminator. A PARAGRAPH is defined as a group of characters bounded on either side by an empty line (two line terminators in succession – displayed by EDT as <CR>). A PAGE is bounded by a form feed, top of buffer, or bottom of buffer.

In resetting delimiters, remember that the SET ENTITY command overwrites the default values. If you want to add delimiters to the existing ones, you must include all the original delimiters in your command string, not just the new ones you want to add. The delimiters you specify must be enclosed in quotation marks (single – ' or double – "). If you want to include both kinds of quotation marks in your list of delimiters, double the kind that is used to enclose the list. The command **SET ENTITY SENTENCE ". ? "" ' ! "** has all the default sentence boundary characters plus both kinds of quotation marks.

The elements in the WORD and SENTENCE boundary list are single characters, each of which is a boundary by itself. You can have any number of different WORD and SENTENCE boundaries at the same time during your EDT session. Do not use spaces or any other characters to separate the individual boundary marks in the SET ENTITY WORD or SENTENCE command.

For PARAGRAPH and PAGE, you can only have a single boundary at any given time, but that boundary can be composed of several characters. For example, your PAGE boundary can be the word **PAGE**. If you use letters in your PARAGRAPH or PAGE boundary, EDT uses its current search parameters when locating the boundary string. Thus, if SET SEARCH EXACT is in effect and the PAGE boundary is **.PAGE**, EDT will not recognize **.page** as a page separator.

For SET ENTITY WORD, EDT considers all word boundaries, except spaces, to be word entities themselves unless **SET WORD NODELIMITER** is in effect.

See the section on the SET ENTITY command in Chapter 6 for more information on how to define entities and how to use them in startup command files and EDT macros.

Examples:

```
*SET ENTITY WORD '        {}[]()<LF><VT><FF><CR>'
```

Sets the word entity to include all the default boundaries as well as braces, brackets, and parentheses.

```
*SET ENTITY SENTENCE ".!?'"")]"
```

Sets the sentence entity to be any string of characters ending with ., !, ?, ', ",), or] followed by at least one space or line terminator. Notice that the double quotation mark (") inside the boundary list is doubled.

```
*SET ENTITY PARAGRAPH "<CR><CR><CR>"
```

Sets the paragraph entity to be any group of lines separated by two empty lines.

```
*SET ENTITY PAGE '.HL1'
```

Sets the page entity to be any text beginning with a RUNOFF level-one header.

Syntax:

```
SET FNF
SET NOFNF
```

Description:

The SET NOFNF command suppresses the message that appears when you use EDT to create a new file. (FNF stands for File Not Found.) Normally, when you call up EDT to create a file, you see this message:

```
Input file does not exist
```

This is the default state – SET FNF.

When you have the SET NOFNF command in an EDT startup command file, EDT does not display the **Input file does not exist** message at the start of your editing session. Note that SET NOFNF does not suppress the message **Input file does not have standard text file format**.

Example: Shows the start of an EDT session in which the "Input" message is displayed and then a session in which the message has been suppressed by placing the SET NOFNF command in the startup command file.

```
☞ EDIT /EDT JUL141989.RNO

Input file does not exist.
[EOB]
*
   .
   .
   .
*EXIT
```

(Now put the SET NOFNF command in your startup command file.)

```
☞ EDIT /EDT JUL151989.RNO

[EOB]
*
```

SET HELP Command
Line

Syntax:

```
SET HELP [file-specification]
```

Description:

The SET HELP command enables you to access different HELP files for your EDT session. Since EDT allows users to modify the HELP file text or create new HELP files, you might have access to more than one HELP file. The SET HELP command enables you to use the HELP file that corresponds to your key definitions and EDT macros, even if that file is not the default HELP text for your site.

If you have key definitions and EDT macros that are documented in a revised HELP file, include the appropriate SET HELP command in your startup command file so that your HELP text will be consistent with your editing environment. See the section in Chapter 7 on creating and modifying EDT's HELP file.

When you issue the SET HELP command without a file specification or type SET HELP EDTHELP, EDT returns you to the default HELP file. The **SHOW HELP** command tells you the name of the current HELP file.

Each system has its own syntax for the HELP file specification. If you omit parts of the file specification in your SET HELP command, the following defaults will be supplied for your operating system:

	Device Name	File Name	File Type
VAX/VMS	SYS$HELP:	EDTHELP	.HLB
RSTS/E	LB:	EDTHEL	.HLP
RSX-11M/M-PLUS	LB:[1,2]	EDTHELP	.HLP

Examples:

```
*SET HELP EDTHELP
```
Specifies EDT's default HELP file on VAX/VMS and RSX-11M/M-PLUS systems.

```
*SET HELP EDTHEL
```
Specifies EDT's default HELP file on RSTS/E.

```
*SET HELP DISK$USER:[SMITH]XEDTHELP.HLB
```
Specifies a different EDT HELP file on VAX/VMS.

```
*SET HELP SY:[300,1]XEDTHP.HLP
```
Specifies a different EDT HELP file on RSTS/E.

```
*SET HELP SY:[30,1]XEDTHELP.HLP
```
Specifies a different EDT HELP file on RSX-11M/M-PLUS.

Syntax:

```
SET KEYPAD
SET NOKEYPAD
```

Description:

The SET KEYPAD/NOKEYPAD command determines which screen mode EDT accesses. The default screen mode is keypad. When you begin an EDT session and give the **CHANGE** command, your editing session is automatically shifted to keypad mode. If you give the SET NOKEYPAD command and then the CHANGE command, you enter nokeypad mode.

If you have the **SET MODE** CHANGE command in your startup command file, EDT automatically starts your session in keypad mode. To have your EDT sessions start in nokeypad mode, you must include the SET NOKEYPAD command in the startup command file.

You can use the SET NOKEYPAD command with the keypad **COMMAND** function to shift directly to nokeypad editing from keypad mode. Conversely, when you are in nokeypad mode, you can issue the **EXT** SET KEYPAD command to shift directly to keypad editing.

Once you have issued a SET NOKEYPAD command, you can switch back and forth between line mode and nokeypad mode without having to reissue the SET NOKEYPAD command.

See the section on shifting editing modes in Chapter 7 for more information about shifting to different editing modes.

Example 1: Shifts from line mode to nokeypad mode at the start of an editing session.

```
☞ EDIT /EDT LETTER.RNO

*SET NOKEYPAD
*CHANGE
```

Example 2: Shifts from line mode to keypad mode when SET NOKEYPAD is in effect.

```
*SET KEYPAD
*CHANGE
```

SET LINES Command
Line

Syntax:

```
SET LINES number
```

Description:

The SET LINES command is used to limit the number of lines that EDT displays on the terminal screen at one time. The number specifier can be any integer from 1 to 22. The default value is the maximum: 22 lines per screen. This command has no effect on the number of lines moved by the keypad **SECT** function. SET LINES also has no effect in line mode.

If you reduce the number of lines on the screen, use the **SET CURSOR** command to adjust the cursor settings that tell EDT when to scroll the screen. If you set the number of lines small enough to invalidate the current SET CURSOR values, EDT adjusts the SET CURSOR values so that they will be valid. However, EDT does not change the SET CURSOR values when you increase the number of lines displayed per screen. Use the **SHOW CURSOR** command to see the current cursor settings.

The SET LINES command is useful if you are working at a terminal operating at a low data transmission rate. With fewer lines displayed on the screen, EDT requires less time to redraw the screen when you issue a command or keypad keystroke that significantly alters the screen image.

Example: Limits the screen display to eight lines. Checks the new cursor setting and then uses the SET CURSOR command to reset the scrolling lines to 2 and 6.

```
This is line 1.
This is line 2.
This is line 3.
This is line 4.
This is line 5.
This is line 6.
This is line 7.
This is line 8.
This is line 9.
This is line 10.
   .
   .
   .
This is line 22.
```

```
EXT SET LINES 8

     This is line 1.
     This is line 2.
     This is line 3.
     This is line 4.
     This is line 5.
     This is line 6.
     This is line 7.
     This is line 8.

EXT SHOW CURSOR
7:7

EXT SET CURSOR 2:6
```

SET MODE Command
Line

Syntax:

```
SET MODE CHANGE
SET MODE LINE
```

Description:

The SET MODE command establishes the initial mode of your EDT session. When SET MODE CHANGE is in an EDT startup command file, your editing session automatically starts off in a change mode instead of line mode, which is the default.

The default change mode is keypad mode if the operating system informs EDT that you have either a VT100-type or VT52 terminal. If you are using some other kind of terminal, the default change mode is hardcopy change mode.

The **SHOW MODE** command shows the mode that was last set by a SET MODE command. The mode that SHOW MODE displays will not necessarily correspond to the current EDT mode since you can use other commands to shift from one editing mode to another during your editing session.

Example: Shows an EDT session starting out in the default mode – line. Then shows the beginning of the same session if SET MODE CHANGE is in the startup command file.

```
☞ EDIT /EDT LETTER.RNO
    1        January 1, 1986
*
```

(After you put the SET MODE CHANGE command in the startup command file, reissue the EDIT /EDT command.)

```
☞ EDIT /EDT LETTER.RNO

January 1, 1986


Mr. Jordan R. Kingsberry
    .
    .
    .
[EOB]
```

Syntax:

```
SET NUMBERS
SET NONUMBERS
```

Description:

SET [NO]NUMBERS determines whether or not EDT displays its line numbers during line mode editing. If you request SET NONUMBERS, EDT displays text in line mode starting at the left margin of your screen or paper, without showing the EDT line numbers. To restore the default state, give the SET NUMBERS command. EDT maintains its line numbering system even when SET NONUMBERS is in effect.

Example: The first TYPE command lists lines 3 through 5 with the EDT line numbers displayed. The second TYPE command lists the same lines when SET NONUMBERS is in effect.

```
*TYPE 3 THRU 5

      3       Ms. Phyllis Davenport
      4       3587 Charter Lane
      5       Hartford, CT  06107

*SET NONUMBERS
*TYPE 3 THRU 5

      Ms. Phyllis Davenport
      3587 Charter Lane
      Hartford, CT  06107
```

SET PARAGRAPH [NO]WPS Command
Line

Syntax:

```
SET PARAGRAPH WPS
SET PARAGRAPH NOWPS
```

Description:

The default boundary limits for a paragraph in EDT are two line terminators in succession. When you use the paragraph entity to move the cursor in a change mode with EDT in the forward direction, EDT positions the cursor on the character immediately following the second line terminator of the paragraph boundary. SET PARAGRAPH NOWPS is the default.

The SET PARAGRAPH WPS command recognizes an empty line as the paragraph boundary, but does not leave the cursor at the next character after the empty line if that character is another line terminator. With SET PARAGRAPH WPS in effect, EDT looks for the first character beyond the paragraph boundary that is not a line terminator and moves the cursor there.

If EDT's direction is backward and SET PARAGRAPH WPS is in effect, EDT moves the cursor to the nearest character preceded by two line terminators in succession, providing that character is not a line terminator itself.

If you have changed the paragraph boundary with the SET ENTITY PARAGRAPH command and the new entity ends with a line terminator (<CR>), SET PARAGRAPH WPS will cause EDT to skip over any blank lines that might follow directly after the paragraph entity.

Example: Shows the cursor position when the PAR entity is used with the default SET PARAGRAPH NOWPS. Then shows the cursor position when SET PARAGRAPH WPS is in effect. The EDT direction is forward.

```
@t any time since the last war.


Later that day she was to regret her unkind remark.
True, it had been unintentional, but she could tell
PAR

at any time since the last war.

0

Later that day she was to regret her unkind remark.
True, it had been unintentional, but she could tell
```

(Move the cursor back to the **a** in **at**.)

```
EXT SET PARAGRAPH WPS

PAR

    at any time since the last war.


    Later that day she was to regret her unkind remark.
    True, it had been unintentional, but she could tell
```

SET PROMPT Command
Line

Syntax:

```
SET PROMPT prompt-type "string"
```

Description:

The SET PROMPT command is used during EDT development to facilitate automatic testing of EDT. It allows redefinition of the prompt string that is displayed in line mode, in keypad mode, in nokeypad mode, in hardcopy change mode, in line mode when inserting, in line mode when inserting with NONUMBERS, and in line mode with the /QUERY qualifier.

Prompt types cannot be abbreviated. They are:

LINE	KEYPAD	NOKEYPAD	HCCHANGE
INSERT	INSERTN	QUERY	

EDT's automatic testing facility uses SET PROMPT to add a unique, nonprinting character (CTRL/A) to the end of each prompt. The character signals the tester that EDT is ready for simulated user input.

Use of this command is not recommended. Note that with SET PROMPT in effect, EDT will become confused if the new prompt string for the keypad or nokeypad prompt causes a net motion of the cursor. Also note that if a <CR><LF> pair is not included in the new string for any prompt that has those characters in the default prompt, these characters will be inserted at the beginning of the new string. The insert prompt is modified if the line numbers near the point of insertion become large.

The default prompts are:

```
LINE:       <CR><LF>*     INSERT:     <CR><LF>
KEYPAD:                   INSERTN:    <CR><LF>
NOKEYPAD:                 QUERY:      <CR><LF>?
HCCHANGE:   <CR><LF>C*
```

Examples:

```
*SET PROMPT LINE "<CR><LF>!!!"
```
Sets the line mode prompt to an end of line followed by three exclamation marks.

```
*SET PROMPT KEYPAD "^@"
```
Sets the keypad mode prompt to a string containing one null character.

```
*SET PROMPT NOKEYPAD ""
```
Sets the nokeypad mode prompt to nothing.

```
*SET PROMPT HCCHANGE "<CR><LF>CHANGE"
```
Sets the hardcopy change mode prompt to an end of line followed by the word CHANGE.

```
*SET PROMPT INSERT "<CR><LF>{"
```
Sets the line mode insert prompt to an end of line followed by a left brace.

```
*SET PROMPT INSERTN "<CR><LF>+"
```
When SET NONUMBERS is in effect, sets the line mode insert prompt to an end of line followed by a plus sign.

```
*SET PROMPT QUERY "<CR><LF>?????"
```
Sets the line mode /QUERY prompt to an end of line followed by five question marks.

SET [NO]QUIET Command
Line

Syntax:

```
SET QUIET
SET NOQUIET
```

Description:

The SET QUIET command silences the terminal bell that ordinarily sounds whenever EDT issues an error message during a screen mode editing session. Use SET NOQUIET, the default, to restore the bell sound.

SET [NO]QUIET has no effect on the nokeypad **BELL** command.

Example: Shows the SET QUIET command in a startup command file.

```
SET SEARCH EXACT
SET QUIET
SET MODE CHANGE
```

Syntax:

```
SET REPEAT
SET NOREPEAT
```

Description:

The SET NOREPEAT command disallows use of the GOLD repeat feature, which enables you to repeat functions in keypad mode. The default is SET REPEAT, where pressing GOLD followed by a keyboard digit (or digits) causes the subsequent keypad function to be repeated as many times as the number entered. SET NOREPEAT also disallows the use of the **SPECINS** keypad function, which enables you to insert any character from the DEC Multinational Character Set into your text by using its decimal equivalent value.

If you are accustomed to a word processing system, you might find the NOREPEAT option more convenient to use.

Example: Shows the SET NOREPEAT command in a startup command file.

```
SET PARAGRAPH WPS
SET NOREPEAT
SET MODE CHANGE
```

SET SCREEN Command
Line

Syntax:

```
SET SCREEN width
```

Description:

The SET SCREEN command allows you to change the maximum number of characters displayed on each screen line during your editing session. The width specifier can be either 80 or 132 for VT100-type terminals (including VT102s) with advanced video option (AVO) or 80 for VT52s and VT100s without AVO (including VT101s). SET SCREEN also affects the number of characters displayed on your paper if you are using a hardcopy terminal. For hardcopy terminals, the width specifier can take any value between 1 and 132.

Be sure not to specify a value for width that exceeds your terminal's capacity or exceeds 132. Otherwise, EDT becomes confused and cannot process your edits properly.

To see if your VT100-type terminal has AVO, press the SET-UP key at the top left corner of the keyboard. The word **SET-UP** is displayed in large letters at the top of the screen. If SET-UP flashes, your terminal has the advanced video option and can accommodate a SET SCREEN value of 80 or 132. Press the SET-UP key again to restore the previous screen display.

The SET SCREEN command does not cause EDT to wrap long lines in the screen modes. Use the **SET NOTRUNCATE** command to show the ends of lines that exceed the screen width. Use **SET WRAP** in keypad mode to have EDT wrap lines of text as you insert them.

You must use the SET WRAP command to change the line width for the keypad **FILL** function and the nokeypad FILL command because SET SCREEN can only accept values of 80 or 132 in screen modes. EDT uses the SET SCREEN value to determine the line length for filling text only if SET NOWRAP (the default) is in effect. If SET WRAP is in effect, EDT always uses the wrap value, regardless of the SET SCREEN width.

In line mode, EDT always wraps lines that are longer than the SET SCREEN width specifier.

EDT gets the initial screen width from the operating system or from commands either in your login file or an EDT startup command file. Remember that the only screen width possible for VT52 terminals is 80 characters. Using a width specifier of 132 on a VT52 causes confusing line wraps and truncations. If you try to use EDT on a VT52 and your working line length is greater than 80, use QUIT to leave EDT and reset whatever is sending EDT that information. Then start your editing session over again. You can also use EDT's SET SCREEN 80 command at the beginning of your session. Use the **SHOW SCREEN** command to determine the current screen width setting.

When a line exceeds the screen width in a screen mode, EDT indicates that there are additional characters on the line by placing a diamond (◇) in the last column. If the number of characters on your line equals the current screen width exactly, you will see all of them. If you have one more character, you will see one less character than the screen width value. The last column will contain the diamond. In other words, if your screen width is 80 and you have exactly 80 characters in the line, all 80 characters will be visible. If you have 81 or more characters in your line, only the first 79 will be visible.

Example: Increases the screen width so that you can see the entire line.

```
This is a line of text that is longer than 80 characters, the screen default wi◇
```

GOLD + PAGE COMMAND

```
Command: SET SCREEN 132
```

ENTER SUBS

```
This is a line of text that is longer than 80 characters, the screen default wi◇th for my VT100.
```

SET SEARCH Command
Line

Syntax:

```
SET SEARCH GENERAL
           EXACT
           WPS
           CASE INSENSITIVE
           DIACRITICAL INSENSITIVE

SET SEARCH BEGIN
           END

SET SEARCH BOUNDED
           UNBOUNDED
```

Description:

The SET SEARCH command influences how EDT locates strings during your editing sessions. All EDT commands that involve search strings, regardless of editing mode, are affected by the SET SEARCH parameters. The keypad mode functions that use search strings are **FIND**, **FNDNXT**, and **SUBS**. In line mode, both substitute commands and all commands that use a string for the range specifier are affected by the SET SEARCH parameters. In nokeypad mode, SET SEARCH influences any command using either the string specifier or the string entity.

There are three groups of SET SEARCH parameters. The first deals with the case and diacritical marks of letters in the search string as compared with the string EDT finds in your text. The second deals with the cursor position after the string has been found. The third determines how much of the text buffer EDT looks at to find a matching string.

1. SET SEARCH {GENERAL | EXACT | WPS | CASE INSENSITIVE | DIACRITICAL INSENSITIVE} determines how EDT matches the case and diacritical marks of the letters in the search string with the letters in the text. When SET SEARCH GENERAL, the default, is in effect, EDT disregards both case and diacritical marks in performing a search. Thus, **FUR**, **FÜR**, **Fur**, **Für**, **fur**, and **für** all match the search string **fur**.

   With SET SEARCH EXACT, the case and diacritical mark of each letter in the search string must exactly match the case and diacritical mark of each corresponding letter in the string that EDT locates. If the search string is **angel**, EDT ignores **ANGEL**, **Angel**, or **angél** in the text.

   SET SEARCH WPS requires that only uppercase letters in the search string match uppercase letters in the text. The search string **The** matches **THE**, **The**, **THe**, or **ThE**, but not **the** or **tHE**.

   SET SEARCH CASE INSENSITIVE matches diacritical marks exactly, but disregards the case of the letters. The words **Eon** and **eon** match the search string **EON**; **Eón** and **eón** do not. Conversely, SET SEARCH DIACRITICAL INSENSITIVE matches the case of letters exactly, but ignores any diacritical marks EDT might encounter. The words **piece** and **pièce** match the search string **piece**; **Piece** and **Pièce** do not.

2. SET SEARCH {BEGIN | END} determines where the cursor stops after EDT finds the search string when you are working in a screen mode. With SET SEARCH BEGIN, the default, EDT positions the cursor at the first character in the found string. With SET SEARCH END, the cursor stops at the character immediately to the right of the last character in the string. This parameter has no effect in line mode.
3. SET SEARCH {BOUNDED | UNBOUNDED} determines how much of the text EDT searches through when trying to find a string. With SET SEARCH UNBOUNDED, the default, EDT starts its search at the current cursor position or current line and continues to search until it either finds the string or reaches the end of the buffer. If the current direction is forward, EDT stops at the bottom of the buffer. If the current direction is backward, the search stops at the top of the buffer.

   When SET SEARCH BOUNDED is in effect, EDT stops the search either when it finds the string or when it reaches a page boundary marker. The default page marker is the form feed (<FF> – CTRL/L). To insert form feeds in your text, simply press CTRL/L at the appropriate locations. If you have used the **SET ENTITY** PAGE command to establish a different page marker, EDT stops at that boundary marker when the search parameter is BOUNDED. If there are no page boundary markers in your buffer, SET SEARCH BOUNDED has the same effect as SET SEARCH UNBOUNDED.

Use the **SHOW SEARCH** command to have EDT display the current search parameters.

Example 1: Shows EDT printing different lines depending on the case of letters in the search string, first with SET SEARCH EXACT and then with SET SEARCH WPS.

```
1       EDT's screen modes use CUT and PASTE to delete, move,
2       and copy text.  Cutting the text removes it from its
3       original location and puts it in a separate buffer.
4       Use PASTE to put a copy of the cut text in a new
5       location.

*SET SEARCH EXACT
*TYPE ALL "CUT"

1       EDT's screen modes use CUT and PASTE to delete, move,

*TYPE ALL "cut"

4       Use PASTE to put a copy of the cut text in a new

*SET SEARCH WPS
*TYPE ALL "Cut"

1       EDT's screen modes use CUT and PASTE to delete, move,
2       and copy text.  Cutting the text removes it from its
```

Example 2: Shows the cursor located on the character after the search string in screen mode (nokeypad in this case).

```
     Maynard, MA  01754

EXT SET SEARCH END

"01754"

     Maynard, MA  01754
```

Example 3: Stops EDT from looking for the string after line 254. Therefore, EDT prints the message **String was not found**.

```
     254        After the long trip home, he needed some rest.
     255        <FF>
     256        The next afternoon, Martha came to the house for tea.

*TYPE .
     254        After the long trip home, he needed some rest.

*SET SEARCH BOUNDED
*TYPE "Martha"

     String was not found
```

Syntax:

```
SET SUMMARY
SET NOSUMMARY
```

Description:

The SET NOSUMMARY command suppresses the summary information printed at the terminal when you use the **EXIT** or **WRITE** command to copy text from your EDT session into an external file.

Normally with the default state, SET SUMMARY, EDT prints a line of information giving you the complete file specification and number of lines in the file that EDT has created as a result of your EXIT or WRITE command. With SET NOSUMMARY, only the appropriate prompt appears after the command has been processed.

Example: Shows the default summary information printed after the EXIT command. Then shows no summary information being supplied when SET NOSUMMARY is in effect.

```
*EXIT

     DISK$USER:[SMITH.SUBCAT1]MEMO35.RNO;3 36 lines
     ☞

☞ EDIT /EDT MEMO35.RNO
    .
    .
    .

*SET NOSUMMARY
*EXIT

     ☞
```

SET TAB Command
Line

Syntax:

```
SET TAB number
SET NOTAB
```

Description:

The SET TAB command establishes the SET TAB value for various tabbing functions in all three editing modes. The default for this command is SET NOTAB. With SET NOTAB in effect only the keypad mode **TAB** (CTRL/H) functions and the nokeypad **TAB** command have any effect on your text. The TAB key also works when you are in line mode to insert horizontal tab characters in your text. The remaining tabbing commands and functions do nothing unless you have issued the SET TAB command.

SET TAB activates EDT's tabbing facility that enables you to format layered text such as outlines and indented computer programs. The keypad tabbing functions are:

CTRL/A	(tab compute)	CTRL/E	(tab increment)
CTRL/D	(tab decrement)	CTRL/T	(tab adjust)

The corresponding nokeypad tabbing commands are:

TC	(tab compute)	TI	(tab increment)
TD	(tab decrement)	TADJ	(tab adjust)

SET TAB also activates the line mode **TAB ADJUST** command.

The SET TAB value establishes the working value for the tabbing facility. The tab compute, decrement, and increment functions use the SET TAB value to determine the number of tab stops for EDT to indent a line. The tab adjust functions enable you to indent blocks of text. See Chapter 7 for a full description of EDT's tabbing capabilities.

Example: After setting the SET TAB value to 5, uses the TAB key to indent the line 5 columns.

```
PLASTIC RINGS
39    TC076    No. 46 Plastic Rings    .03
```

GOLD + PAGE COMMAND

```
Command:  SET TAB 5
```

ENTER SUBS

TAB

```
PLASTIC RINGS
     39    TC076    No. 46 Plastic Rings    .03
```

Syntax:

```
SET TERMINAL HCPY
             VT100
             VT52

SET TERMINAL SCROLL
             NOSCROLL

SET TERMINAL EIGHTBIT
             NOEIGHTBIT

SET TERMINAL EDIT
             NOEDIT
```

Description:

With the SET TERMINAL command, you can correct or change EDT's assumptions about the type of terminal you are using. Generally, EDT is able to get all the information it needs to know about your terminal from the operating system. However, in some instances the data it receives is incorrect. At other times, you might want to change one or more of the settings for some special purpose.

Note that you must *not* issue a SET TERMINAL {VT100 | VT52 | HCPY} command from a screen mode. You can issue the other SET TERMINAL commands from any mode (for example, SET TERMINAL NOSCROLL).

There are four terminal characteristics that you can reset:

basic terminal type:	HCPY, VT100, VT52
scrolling regions:	SCROLL, NOSCROLL
additional character display:	EIGHTBIT, NOEIGHTBIT
enhanced screen editing:	EDIT, NOEDIT

If EDT does not support your terminal for screen editing, the terminal should be set for HCPY (hardcopy). This setting allows you to use both line mode and hardcopy change mode, but not the screen modes. (Check Appendix C for the list of terminals that are supported for screen editing.)

The VT100 setting is for terminals belonging to the VT100 family. These terminals have reverse video display and 22 keypad keys. VT100-type terminals include VT101, VT102, VT125, VT131, and VT132 and terminals that have the LK201 keyboard and VT100 emulation.

The VT52 setting is for VT52 terminals. VT52 terminals have 19 keypad keys and do not have reverse video display.

The SCROLL option shows that your terminal has scrolling regions that EDT can use. This is the default for most VT100-type terminals.

If you do not have a VT100-type terminal, be sure that EDT has your terminal set to NOSCROLL. This is the default for VT52 terminals.

The EIGHTBIT option is for terminals that can display characters requiring eight bits for their representation. The eight bit characters are those in the DEC Multinational Character Set with decimal values of 128 through 255. If your terminal has eightbit character capacity, EDT can display the additional printing characters on your screen in their proper graphic representation, for example, ñ for decimal value 241.

If your terminal cannot display the additional DEC Multinational Character Set characters, you need to use the NOEIGHTBIT option. On NOEIGHTBIT terminals, the characters from 128 through 225 are displayed using symbols inside angle brackets, for example <A"> for decimal 196. You need to use the keypad **SPECINS** function or the **ASC** command in either nokeypad or hardcopy change mode to insert these characters in your text. The DEC Multinational Character Set is listed in Appendix G.

The EDIT option is for terminals, such as the VT102, that have internal screen editing features. These features are: IL (insert line), DL (delete line), the insert state of IRM (insertion-replacement mode), and DCH (delete character). If you have a VT102 or similar terminal and it is set to EDIT, EDT is able to use the terminal's internal screen editing features to enhance performance. Note that terminals using the LK201 keyboard have internal screen editing features. The NOEDIT option is for all terminals that do not have the VT102-type internal screen editing features.

The SET TERMINAL command can be used to turn off features that your terminal has, but it cannot be used to increase the capabilities of your terminal. For instance, if your terminal does not have internal editing capabilities, issuing the SET TERMINAL EDIT command merely confuses EDT and causes problems in your editing session.

Example 1: Changes a VT100 or VT52 terminal to the hardcopy setting.

```
*SET TERMINAL HCPY

*SHOW TERMINAL
Hardcopy noscroll noeightbit noedit
```

Example 2: Changes the EDIT/NOEDIT setting to accommodate a VT102 terminal.

```
*SET TERMINAL EDIT

*SHOW TERMINAL
VT100 scroll noeightbit edit
```

Syntax:

```
SET TEXT END "string"
SET TEXT PAGE "string"
```

Description:

The SET TEXT command allows you to personalize two items during the course of your EDT session. SET TEXT PAGE displays the string you supply for any form feed characters (<FF>) in the buffer. SET TEXT END causes EDT to display your string instead of the [EOB] (end of buffer) mark at the end of each EDT buffer. Once you exit from EDT, these strings are not part of the file.

You can put the SET TEXT END and SET TEXT PAGE commands in a startup command file or you can issue them during your editing session. They have no permanent effect on the text that you are editing.

The strings used for PAGE and END can contain any printing characters (control characters and DELETE are not allowed). These strings can have several words, but cannot exceed a single line.

Example: Changes the characters that EDT prints out for <FF> and [EOB].

```
<FF>
I will be in touch with you as soon as I have found a
source for the plastic tubing.

Sincerely yours,
  .
  .
  .
[EOB]

EXT SET TEXT PAGE "PAGE BREAK HERE!"

EXT SET TEXT END "END OF LETTER, DID YOU REMEMBER THE CCs?"

PAGE BREAK HERE!
I will be in touch with you as soon as I have found a
source for the plastic tubing.

Sincerely yours,
  .
  .
  .
END OF LETTER, DID YOU REMEMBER THE CCs?
```

SET [NO]TRUNCATE Command
Line

Syntax:

```
SET TRUNCATE
SET NOTRUNCATE
```

Description:

The SET NOTRUNCATE command causes lines longer than the current screen width to wrap onto subsequent lines when you are working in a screen mode. The default screen width is the terminal width that the operating system reports to EDT at the start of your editing session. This width can be changed by the **SET SCREEN** command.

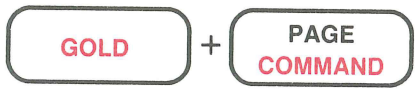
When you use SET NOTRUNCATE, EDT marks the start of the wrapped portion of a line with a diamond followed by a space. (SET NOTRUNCATE has no effect when you are working in line mode, thus no diamond appears on the screen or paper. In line mode, EDT always wraps long lines.)

SET NOTRUNCATE uses only the screen width value to determine where to break the line. It does not take word boundaries into consideration. Use **SET WRAP** to break lines at word boundaries when you insert text in keypad mode.

SET NOTRUNCATE has no effect on the text in your buffer, only on how that text is displayed on the screen. SET WRAP has a permanent effect on your text.

Example: Shows the difference between the text when SET TRUNCATE is in effect, and again when SET NOTRUNCATE is in effect. The SET SCREEN width is 80.

```
We are now planning our new sales campaign for the fall.  I want every sales re◊
to become fully aware of all the products being marketed by our competition.
```




```
Command: SET NOTRUNCATE
```

```
We are now planning our new sales campaign for the fall.  I want every sales rep
◊ resentative
to become fully aware of all the products being marketed by our competition.
```

Syntax:

```
SET VERIFY
SET NOVERIFY
```

Description:

The SET VERIFY command instructs EDT to print at the terminal the commands in a startup command file or EDT macro as the commands are being processed. Under the default condition, SET NOVERIFY, EDT does not display the contents of the startup command file or macro as it is being processed.

Example: First shows the startup command file with SET VERIFY at the top. Then shows the start of your EDT session.

Suppose this is your startup command file:

```
SET VERIFY
FIND =TABLE
INCLUDE TABLE.MAC
FIND =MAIN
SET MODE CHANGE
```

Now start to edit a new file.

```
☞ EDIT /EDT TABLE35.RNO

    FIND =TABLE
    INCLUDE TABLE.MAC
    FIND =MAIN
    SET MODE CHANGE

    [EOB]
      .
      .
      .
    Input file does not exist
```

SET WORD [NO]DELIMITER Command
Line

Syntax:

```
SET WORD DELIMITER
SET WORD NODELIMITER
```

Description:

The SET WORD NODELIMITER command changes the way EDT responds to word entity boundaries. Normally, EDT considers all word delimiters except the space to be words themselves when deleting words or moving the cursor a word at a time. After you issue the SET WORD NODELIMITER command, EDT still uses the word delimiters to determine where words begin and end, but it does not treat these characters as separate words. SET WORD DELIMITER is the default.

Example: First shows the cursor moving to the line terminator when SET WORD DELIMITER is in effect. Then, using the same sample text, shows the cursor moving to the first character of the next line with SET WORD NODELIMITER in effect.

```
During the winter months, people who live in the northern
states like to travel to the south for vacations as well
as business trips.
```

WORD CHNGCASE + WORD CHNGCASE

```
During the winter months, people who live in the northern
states like to travel to the south for vacations as well
as business trips.
```

(Now repeat the example with SET WORD NODELIMITER in effect, first moving the cursor back to the **t** in **the northern**.)

GOLD + PAGE COMMAND

```
Command: SET WORD NODELIMITER
```

ENTER SUBS

WORD CHNGCASE + WORD CHNGCASE

```
During the winter months, people who live in the northern
States like to travel to the south for vacations as well
as business trips.
```

Syntax:

```
SET WRAP number
SET NOWRAP
```

Description:

The SET WRAP command causes lines of text to wrap when new text is being inserted into a buffer in keypad mode. The command also determines the line length for the keypad **FILL** function and the line and nokeypad mode **FILL** commands. The number specifier tells EDT what the maximum line length should be for the inserted or filled text. SET NOWRAP is the default.

Two important things to remember about SET WRAP are: (1) that it has no effect on text already in your buffer, and (2) that it has no effect on text that you insert in line or nokeypad mode.

When you are inserting text in keypad mode, SET WRAP always breaks the line at a space, line terminator or other word boundary so that no words are divided. If there are no spaces in the line, you can type up to 255 characters on a line and still have no wrapping take place, regardless of the SET WRAP value. If you are inserting characters in the middle of an existing line in such a way that no spaces are added, no break occurs, even if there are spaces elsewhere in the line.

SET WRAP has no effect on lines that are already in a buffer, but once set, it has a permanent effect on text that is inserted in keypad mode or text that has been reformated by a FILL function. When a wrap takes place, EDT considers the wrapped portion to be a separate line and numbers it accordingly. So, although *you* did not enter a line terminator, one now exists at the spot where the wrap was made. The wrapped portion of the line has its own line number and maintains its separateness during your editing session and in any external files that EDT creates from that text.

When you use SET WRAP with FILL, the number specified for the SET WRAP command is used by EDT to determine the maximum line length. If SET NOWRAP is in effect, the line length for FILL is determined by the **SET SCREEN** value. However, SET WRAP overrides SET SCREEN.

In screen mode, the **SET NOTRUNCATE** command causes EDT to wrap all lines that exceed the current SET SCREEN setting. SET NOTRUNCATE breaks the line at the SET SCREEN width, regardless of word boundaries. NOTRUNCATE has no permanent effect on your text.

Example 1: Before entering keypad mode, sets the SET WRAP value to 40. EDT then wraps the text as you insert it.

```
*SET WRAP 40
*CHANGE
```

(Here is the line as you start to type it.)

```
     As we type this line, we pause just befo
```

(Here is the text as it appears when you finish.)

```
     As we type this line, we pause just
     before the 41st character position.
```

Example 2: Sets the SET WRAP value to 40 and then uses the FILL nokeypad command to reformat the text.

```
     You can devise command procedures to simplify and
     enhance your program development.
     For example,
     you can write a command procedure that will
     compile, link, and run a specific
     PL/I program.

EXT SET WRAP 40

FILL

     You can devise command procedures to
     simplify and enhance your program
     development. For example, you can write
     a command procedure that will compile,
     link, and run a specific PL/I program.
```

Syntax:

```
[count]SHL
```

Description:

The SHL (shift left) command moves the entire buffer text eight characters (one tab stop) to the left. The first eight characters on each line no longer appear on the screen. Assuming that your screen width is set to 80, all characters in positions 80 through 87 that were previously truncated are now visible.

No characters are actually added to or deleted from the text when you use the SHL and **SHR** commands. The effects of the commands are merely visual and temporary. On a VT100 terminal with advanced video option, or on a VT102, you can use the **SET SCREEN** command to display up to 132 characters on a single line, but the size of the characters is reduced.

The count specifier designates the number of tab stops to shift left. When count is 4, the buffer is shifted 32 character positions to the left. Use the SHR (shift right) command to move the buffer contents to the right, back to the original position.

You can define the GOLD/← key sequence on VT100-type terminals to perform the SHL function in keypad mode. The following **DEFINE KEY** command can be included in your startup command file or issued during your editing session:

```
DEFINE KEY GOLD 15 AS "SHL."
```

Example: Shows the effect of SHL on a line of text that has been truncated because it exceeds the screen width of 80.

```
AAAAAAAbbbbbbbbCCCCCCCCddddddddFFFFFFFFggggggggHHHHHHHHiiiiiiiiJJJJJJJJKKKKKKK◇

SHL

bbbbbbbbCCCCCCCCddddddddFFFFFFFFggggggggHHHHHHHHiiiiiiiiJJJJJJJJKKKKKKKKlllllllll
```

SHOW AUTOREPEAT Command
Line

Syntax:

```
SHOW AUTOREPEAT
```

Description:

This command indicates whether **SET AUTOREPEAT** or SET NOAUTOREPEAT is currently in effect. The default for EDT is generally AUTOREPEAT. Use SET NOAUTOREPEAT to change the default if your terminal requires that setting.

Example: Prints the current autorepeat setting.

```
*SHOW AUTOREPEAT
autorepeat
```

Syntax:

```
SHOW BUFFER
```

Description:

The SHOW BUFFER command lists all accessible buffers currently in your EDT session. Inaccessible storage areas, such as the delete character, delete word, delete line, search, and substitute buffers, do not appear in the SHOW BUFFER list. EDT uses the buffer specifier in both line mode and nokeypad mode to create buffers.

When you issue the SHOW BUFFER command, EDT indicates the current buffer by preceding its name with an equal sign (=). EDT also lists the number of lines in each buffer. If the number of lines in the MAIN buffer is marked with an asterisk (*), it means that EDT has not read through the entire buffer and therefore is only aware of the number of lines shown. You can have EDT "read" the entire buffer by issuing a command such as **TYPE** END.

Once a buffer has been created, it remains open for the remainder of the EDT session, unless it is deleted by the **CLEAR** command. The MAIN and PASTE buffers are automatically created by EDT at the start of every editing session. These buffers cannot be eliminated by the CLEAR command, although their contents can be deleted.

Example: First shows the current buffer status at the start of your EDT session, when EDT has seen only the first line of the MAIN buffer. Then shows the buffer status after you have used the TYPE command to move to the end of the buffer.

```
☞ EDIT /EDT LETTER.RNO

     1          March 4, 1984

*SHOW BUFFER

     =MAIN    1*        lines
      PASTE   No        lines

*TYPE END

     [EOB]

*SHOW BUFFER

     =MAIN    45        lines
      PASTE   No        lines
```

SHOW CASE Command
Line

Syntax:

```
SHOW CASE
```

Description:

The SHOW CASE command tells you which case, if any, has been established by the **SET CASE** command. The possible responses are: Upper, Lower, or None. None is the default.

Example: Shows the current case setting.

```
*SHOW CASE
None
```

Syntax:

```
SHOW COMMAND
```

Description:

The SHOW COMMAND command prints the name of the active startup command file. Include SHOW COMMAND in the startup command file to have EDT print the name of that file at the start of your EDT session.

When the **SET COMMAND** command appears in a startup command file, it causes EDT to look for another startup command file. You can include a SHOW COMMAND command in each startup command file so that you will see which files have been processed before you begin your editing session. If you issue the SHOW COMMAND command during your editing session, EDT displays nothing.

Examples: In this example, each SHOW COMMAND command is in a separate startup command file. The file name that EDT prints appears as soon as EDT completes processing that startup command file.

(First startup command file)

```
      .
      .
      .
SET MODE CHANGE
SHOW COMMAND
SET COMMAND EDTEXPERT
```

(Second startup command file)

```
      .
      .
      .
DEFINE KEY GOLD E AS "EXT EXIT."
SHOW COMMAND
```

```
☞ EDIT/EDT ACCTPROG.COB

    EDTSYS
    EDTEXPERT
        .
        .
        .
```

SHOW CURSOR Command
Line

Syntax:

```
SHOW CURSOR
```

Description:

The SHOW CURSOR command displays the values that have been set by the **SET CURSOR** command. The SET CURSOR command establishes the cursor positions at which scrolling of the screen image occurs. The default values are 7:14 – that is, the screen image scrolls up when the cursor is on the 15th line of the screen and you are moving forward. The screen image scrolls down when the cursor is on the 8th line of the screen and you are moving backward. (The SET CURSOR values for the 22 lines on the screen range from 0 to 21.)

Example: First shows the default cursor values. Then after a new set has been established, shows the new values.

```
*SHOW CURSOR
7:14

    .
    .
    .

*SET CURSOR 2:20

    .
    .
    .

*SHOW CURSOR
2:20
```

Syntax:

```
SHOW ENTITY WORD
SHOW ENTITY SENTENCE
SHOW ENTITY PARAGRAPH
SHOW ENTITY PAGE
```

Description:

The SHOW ENTITY command lists the current delimiters that determine the boundaries of four entities: WORD, SENTENCE, PARAGRAPH, and PAGE. Use the **SET ENTITY** command to change the boundaries for these entities.

Examples: These examples show the default boundary settings for each entity.

```
*SHOW ENTITY WORD
        <LF><VT><FF><CR>

*SHOW ENTITY SENTENCE
.!?

*SHOW ENTITY PARAGRAPH
<CR><CR>

*SHOW ENTITY PAGE
<FF>
```

SHOW FILES Command
Line

Syntax:

```
SHOW FILES
```

Description:

The SHOW FILES command displays the current input file and output file for your EDT session. The input file is displayed exactly as you entered it on the EDIT/EDT command line. If you use either the /OUTPUT qualifier or output-file= specifier in the EDIT/EDT command line, SHOW FILES displays the output file exactly as you typed it. If you do not include an output file specification in the command line, SHOW FILES displays the same information for both the input file and the output file.

Example 1: Displays the input and output files for an EDT session when no output file is specified in the EDIT/EDT command line.

```
☞ EDIT /EDT DATALIST.DAT

*SHOW FILES

Input  File:  DATALIST.DAT
Output File:  DATALIST.DAT
```

Example 2: Displays the input and output files for an EDT session under DCL.

```
☞ EDIT /EDT /OUTPUT=SMITH.RNO LETTER.RNO

*SHOW FILES

Input  File:  LETTER.RNO
Output File:  SMITH.RNO
```

Example 3: Displays the input and output files for an EDT session under MCR or CCL.

```
☞ EDIT /EDT LINEMEMO.XXX= [21,33]LINEMEMO.XXX

*SHOW FILES

Input   File:  [21,33]LINEMEMO.XXX
Output  File:  LINEMEMO.XXX
```

Syntax:

SHOW FNF

Description:

The SHOW FNF command indicates whether **SET FNF** or SET NOFNF is in effect. (FNF stands for File Not Found.) The default is SET FNF, meaning that the message **Input file does not exist** is displayed whenever you create a new file with EDT. Put the SET NOFNF command in your startup command file to suppress that message. SET NOFNF does not suppress the message **Input file does not have standard text file format**.

Example: Shows the current FNF setting.

```
*SHOW FNF
nofnf
```

SHOW HELP Command
Line

Syntax:

```
SHOW HELP
```

Description:

The SHOW HELP command tells you which EDT HELP file is currently available for your editing session. The default HELP files are:

VAX/VMS	SYS$HELP:EDHELP.HLB
RSTS/E	LB:EDTHEL.HLP
RSX-11M/M-PLUS	LB:[1,2]EDTHELP.HLP

If you or your site has created a customized HELP file, and you are not sure which HELP file version is currently in use, you can use the SHOW HELP command to find out that information. Use the **SET HELP** command to access a different HELP file.

Example: First shows the default system HELP file. After you issue a SET HELP command to access a different HELP file, shows the new current HELP file.

```
*SHOW HELP
Help file name:  SYS$HELP:EDTHELP.HLB;1

*SET HELP SYS$HELP:MYEDTHELP.HLB

   .
   .
   .

*SHOW HELP
Help file name:  SYS$HELP:MYEDTHELP.HLB;3
```

Syntax:

```
SHOW KEY [GOLD] keypad-key-number
SHOW KEY GOLD character
SHOW KEY [GOLD] CONTROL character
SHOW KEY [GOLD] DELETE
SHOW KEY [GOLD] FUNCTION key-number
```

Description:

The SHOW KEY command tells you the definition of any keys that have keypad editing functions. It shows you the definition for any key that EDT has preset, such as the keypad keys and the preset control keys, as well as any key that you have defined or redefined during your editing session. (Once you have redefined a preset key, SHOW KEY shows only the new definition.)

Because keypad mode function keys are defined in terms of nokeypad commands, the definition that is displayed for the SHOW KEY command uses the nokeypad command syntax. For example, when you ask for the definition of the CONTROL A key, EDT prints **TC.** in response.

All preset key definitions, except **RESET** and **GOLD**, end with a period. The period indicates that the operation is set into motion as soon as you press the key or key sequence. The RESET and GOLD function definitions are not nokeypad commands; they are special EDT keywords. For a keypad key to have the RESET function, it must be defined exactly as RESET with no period. The same is true for GOLD. Neither RESET nor GOLD can be part of a multicommand definition.

When using the SHOW KEY command you must type the words GOLD and CONTROL in your commands. If you are looking for the definition of the DELETE key, you must type the word DELETE. To find out the definition of a function key on the LK201 keyboard, type the word FUNCTION.

When you want to find out the definition for the **BACKSPACE**, **LINEFEED**, or **TAB** key, you must use the key's CONTROL equivalent: CONTROL H for BACKSPACE, CONTROL J for LINEFEED, and CONTROL I for TAB. For terminals with LK201 keyboards, you need to use the FUNCTION numbers for the F12 and F13 keys: FUNCTION 24 and FUNCTION 25.

Keypad-key-number refers to EDT's numeric designations for the keypad keys. For example, the period key on the keypad has number 16, the ENTER key has number 21, and the down arrow key has number 13. Figure 7-1 gives the keypad key numbers for each key on both the VT100 and VT52 keypads.

FUNCTION key refers to the additional keys on the LK201 keyboard that you can define. These include the six keys on the terminal's "editing" keypad, located above the arrow keys, as well as the keys on the function key row from F6 through F20. You cannot define keys F1 through F5 on the function key row. EDT uses its own numbers for these keys ranging from FUNCTION 1 through FUNCTION 99 to encompass the special user defined keys (UDKs) that are allowed on terminals with LK201 keyboards. Figure 7-3 shows the FUNCTION key numbers for the "editing" keypad keys as well as the function key row.

Example: First shows the default definition for the CTRL/N key sequence. Then shows the new definition after CTRL/N has been defined with the DEFINE KEY command.

```
*SHOW KEY CONTROL N
No definition

*DEFINE KEY CONTROL N AS "?'Nokeypad Command: '."

*SHOW KEY CONTROL N
?'Nokeypad Command: '.
```

Other Examples:

```
*SHOW KEY 7
PAGETOP.
```

Shows the default definition for the 7 key on the keypad.

```
*SHOW KEY 16
SEL.
```

Shows the default definition for the period (.) key on the keypad.

```
*SHOW KEY GOLD 16
RESET
```

Shows the default definition for GOLD + . keypad key sequence.

```
*SHOW KEY CONTROL U
DBL.
```

Shows the default definition for the CTRL/U key sequence.

```
*SHOW KEY DELETE
D-C
```

Shows the default definition for the DELETE key.

```
*SHOW KEY 20
GOLD
```

Shows the default definition for the GOLD key.

```
*SHOW KEY FUNCTION 5
(-16L).
```

Shows the default definition for the Prev Screen key on the LK201 keyboard editing keypad.

```
*SHOW KEY FUNCTION 24
BL.
```

Shows the default definition for the F12 key on the LK201 function key row.

Syntax:

```
SHOW KEYPAD
```

Description:

The SHOW KEYPAD command tells you which screen editing mode is in effect: KEYPAD or NOKEYPAD. Use the **SET [NO]KEYPAD** command to reset your session to a different screen mode. For more information on shifting modes in EDT, see Chapter 7.

Example: Shows the current screen mode setting from line mode when SET NOKEYPAD is in effect.

```
*SHOW KEYPAD
nokeypad
```

SHOW LINES Command
Line

Syntax:

```
SHOW LINES
```

Description:

The SHOW LINES command tells you the number of lines that EDT is displaying on the screen at one time. The default (and maximum) number is 22. Rather than count the number of lines being displayed, you can use SHOW LINES. The **SET LINES** command resets the number of lines per screen. The number of lines ranges from 1 to 22. The **SET CURSOR** command resets the cursor positions that activate the scrolling mechanism.

Example: First shows the default setting for LINES. Then shows the new value after SET LINES 10 is in effect.

```
*SHOW LINES
22

*SET LINES 10

    .
    .
    .

*SHOW LINES
10
```

Syntax:

```
SHOW MODE
```

Description:

The SHOW MODE command tells you which **SET MODE** command was most recently issued. The response does not necessarily correspond to your current editing mode if you have used the line mode **CHANGE** command, the nokeypad **EX** command, or the keypad **CTRL/Z** function to shift editing modes.

Example: Shows that SET MODE CHANGE was the most recent SET MODE command.

```
Command: SHOW MODE
```

```
Command: SHOW MODE
Change
```

SHOW NUMBERS Command
Line

Syntax:

```
SHOW NUMBERS
```

Description:

The SHOW NUMBERS command tells you if the EDT line numbers are being displayed in line mode. When SET NONUMBERS is in effect, EDT does not display the line numbers in line mode. Your text begins at the left margin of your screen or paper. However, even when you have used SET NONUMBERS, EDT continues to keep track of the EDT line numbers. Thus, you can still use them in line mode range specifiers. The default is **SET NUMBERS**.

Example: Shows that SET NONUMBERS is in effect.

```
*SHOW NUMBERS
nonumbers
```

Syntax:

```
SHOW PARAGRAPH
```

Description:

The SHOW PARAGRAPH command tells you whether **SET PARAGRAPH WPS** or SET PARAGRAPH NOWPS is currently in effect for your editing session. SET PARAGRAPH NOWPS is the default. SET PARAGRAPH WPS causes EDT to position the cursor on the first character after the default paragraph boundary that is not a line terminator.

Example: Shows that SET PARAGRAPH NOWPS is in effect.

```
*SHOW PARAGRAPH
nowps
```

SHOW PROMPT Command
Line

Syntax:

```
SHOW PROMPT prompt-type
```

Description:

The SHOW PROMPT command tells you which prompt settings are currently in effect. The **SET PROMPT** commands are used with EDT's testing facility to signal that EDT is ready for simulated user input.

The prompt types cannot be abbreviated. They are:

LINE	KEYPAD	NOKEYPAD	HCCHANGE
INSERT	INSERTN	QUERY	

Examples: These examples show all the default SET PROMPT settings.

```
*SHOW PROMPT LINE
<CR><LF>*

*SHOW PROMPT KEYPAD


*SHOW PROMPT NOKEYPAD


*SHOW PROMPT HCCHANGE
<CR><LF>C*

*SHOW PROMPT INSERT
<CR><LF>

*SHOW PROMPT INSERTN
<CR><LF>

*SHOW PROMPT QUERY
<CR><LF>?
```

Syntax:

```
SHOW QUIET
```

Description:

The SHOW QUIET command tells you if the bell, which sounds when EDT returns an error message in screen mode, has been turned off. With **SET QUIET** in effect, the terminal bell does not sound when EDT issues an error message in one of the screen modes. SET NOQUIET is the default. SET QUIET has no effect on the nokeypad BELL command.

Example: First shows the default setting. Then shows the new setting after SET QUIET is in effect.

```
*SHOW QUIET
noquiet

*SET QUIET

    .
    .
    .

*SHOW QUIET
quiet
```

SHOW REPEAT Command
Line

Syntax:

```
SHOW REPEAT
```

Description:

The SHOW REPEAT command lets you know whether you can use the GOLD repeat feature and the **SPECINS** function in keypad mode. **SET REPEAT** is the default. With SET NOREPEAT, you cannot use GOLD and a digit to repeat a keypad mode function nor can you use the SPECINS function.

Example: Shows the current REPEAT setting.

```
*SHOW REPEAT
repeat
```

Syntax:

```
SHOW SCREEN
```

Description:

The SHOW SCREEN command tells you the current screen width setting. The default screen width is the terminal's width as determined by the operating system. For VT100-type terminals and VT52s the default width is usually 80. VT52 terminals and VT100-type terminals without advanced video option (AVO) cannot accommodate screen widths greater than 80 characters. The only valid screen width settings for screen mode editing are 80 and 132. Line mode can use a screen width value from 1 to 132. Use the **SET SCREEN** command to change the screen width.

Example: First shows the current screen width. Then shows the new screen width after SET SCREEN 132 is in effect.

```
*SHOW SCREEN
80

*SET SCREEN 132

    .
    .
    .

*SHOW SCREEN
132
```

SHOW SEARCH Command
Line

Syntax:

SHOW SEARCH

Description:

The SHOW SEARCH command tells you the current search parameters that EDT uses to locate strings in your text. There are three sets of parameters: (1) those that deal with the case or diacritical marks of letters, (2) those that determine the position of the cursor after the search is completed, and (3) those that determine what portion of the buffer EDT will search to find the string.

The first set contains GENERAL (the default), EXACT, WPS, CASE INSENSITIVE (CI), AND DIACRITICAL INSENSITIVE (DI). GENERAL means that EDT ignores case differences and diacritical marks in searching for strings. EXACT means that all case differences and diacritical marks are considered when matching the search string. WPS means that the case of uppercase letters must be matched exactly when locating search strings. With CASE INSENSITIVE, case is ignored, but diacritical marks are not. Conversely, with DIACRITICAL INSENSITIVE, diacritical marks are ignored, but case differences are not.

The second set of parameters, BEGIN and END, determines the new cursor position after EDT locates the string in a screen mode. With BEGIN, the default, the cursor is on the first character of the string. END puts the cursor on the character to the right of the last string character.

The third set of parameters is BOUNDED or UNBOUNDED. UNBOUNDED, the default, means that EDT searches from the current cursor position to the top or bottom of the buffer, depending on the direction of the search. BOUNDED limits the search from the current cursor position to the next PAGE boundary mark that EDT encounters in the current direction. If there are no PAGE markers in your text, the effect of BOUNDED and UNBOUNDED is the same. The default PAGE marker is the form feed (<FF> – CTRL/L). Use the **SET ENTITY** PAGE command to change the PAGE boundary marker.

The search parameters are used in keypad mode with the **FIND**, **FNDNXT**, and **SUBS** functions. In nokeypad mode, the search parameters are used with the string entity, the string specifier, and the substitute commands. Line mode uses search parameters with substitute operations as well as with the string range specifier.

Example: First shows the default search settings. Then shows the new settings after SET SEARCH EXACT and SET SEARCH END are in effect.

```
*SHOW SEARCH
general begin unbounded

*SET SEARCH EXACT
*SET SEARCH END
    .
    .
    .
*SHOW SEARCH
exact end unbounded
```

SHOW SUMMARY Command
Line

Syntax:

```
SHOW SUMMARY
```

Description:

The SHOW SUMMARY command tells you whether the **SET SUMMARY** feature is in effect for your editing session. If the SET NOSUMMARY command has been given, EDT does not display the summary file information after processing either an **EXIT** or **WRITE** command. SET SUMMARY is the default.

Example: First uses the SET NOSUMMARY command to change the default. Then shows the current SUMMARY setting. Finally shows the effects of SET NOSUMMARY when you issue the EXIT command.

```
*SET NOSUMMARY

    .
    .
    .
*SHOW SUMMARY
nosummary

    .
    .
    .

*EXIT
```

Syntax:

```
SHOW TAB
```

Description:

The SHOW TAB command tells you the current **SET TAB** value (or size) as well as the current tab indentation level count. If no tab value is currently in use, EDT prints **notab** in response to the SHOW TAB command. NOTAB is the default. Use the SET TAB command to establish a tab value.

In order for the following commands and functions to have any effect on your text, you must establish a SET TAB value:

Keypad		*Nokeypad*	*Linemode*
CTRL/A	(tab compute)	TC	
CTRL/D	(tab decrement)	TD	
CTRL/E	(tab increment)	TI	
CTRL/T	(tab adjust)	TADJ	TAB ADJUST

The tab level count displayed by the SHOW TAB command is not affected by any of the following:

- The **n** specifier in the line mode TAB ADJUST command
- A count specifier with the nokeypad TADJ command
- A repeat count used with the keypad CTRL/T function

Example 1: Shows the default TAB setting.

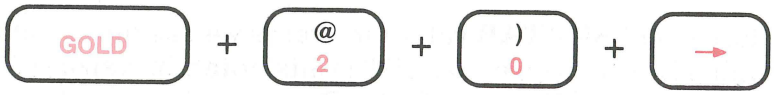
```
*SHOW TAB
notab
```

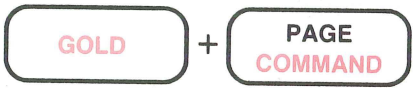
Example 2: Shows the TAB setting after SET TAB 10 is in effect.

```
*SET TAB 10

    .
    .
    .

*SHOW TAB
tab size 10; tab level 1
```

Example 3: First sets the SET TAB value to 5 and shifts to keypad mode. Then uses CTRL/A to reset the tab level count. Shows the new tab size and tab level.

```
*SET TAB 5
*CHANGE
```






```
Command:  SHOW TAB
```

```
Command:  SHOW TAB
tab size 5; tab level 4
```

Syntax:

```
SHOW TERMINAL
```

Description:

The SHOW TERMINAL command shows the terminal settings that are currently in effect for your editing session. Normally, EDT receives the terminal settings from the operating system when you start your editing session. You can use the **SET TERMINAL** command to change these default settings if EDT is receiving incorrect or insufficient information about the terminal you are using.

There are four groups of terminal settings. The first group indicates the terminal type: VT100, VT52, HCPY (hardcopy). The second group refers to scrolling regions in your terminal: SCROLL, NOSCROLL. The third group refers to the number of bits that your terminal uses to represent an individual character: EIGHTBIT, NOEIGHTBIT. The fourth group refers to internal editing features built into the terminal: EDIT, NOEDIT.

Examples: Shows terminal settings for a variety of terminals.

General VT100-type terminal with advanced video option

```
*SHOW TERMINAL
VT100 scroll noeightbit noedit
```

VT52 terminal

```
*SHOW TERMINAL
VT52 noscroll noeightbit noedit
```

Hardcopy terminal or terminal not supported for screen mode editing.

```
*SHOW TERMINAL
Hardcopy noscroll noeightbit noedit
```

VT102 terminal

```
*SHOW TERMINAL
VT100 scroll noeightbit edit
```

SHOW TEXT Command
Line

Syntax:

```
SHOW TEXT END
SHOW TEXT PAGE
```

Description:

The SHOW TEXT command tells you what text EDT is currently displaying for the <FF> page mark or the [EOB] (end of buffer) mark. Use the **SET TEXT** PAGE and SET TEXT END commands to customize these messages. Neither the SET TEXT END command nor the SET TEXT PAGE command has any effect on the text you are editing.

Example: Shows the new text that EDT will display in place of [EOB] and <FF> after you have issued the SET TEXT commands.

```
*SET TEXT END "No more lines in this buffer."
*SET TEXT PAGE "Oh no, not another page!"

    .
    .
    .

*SHOW TEXT END
No more lines in this buffer.

    .
    .
    .

*SHOW TEXT PAGE
Oh no, not another page!
```

Syntax:

```
SHOW TRUNCATE
```

Description:

The SHOW TRUNCATE command tells you whether **SET TRUNCATE** or SET NOTRUNCATE is currently in effect for your editing session. The default is SET TRUNCATE. In screen editing, NOTRUNCATE causes EDT to display the portion of a line that exceeds the current screen width on the next line. SET NOTRUNCATE does not add line terminators to your text. Use **SET WRAP** to add line terminators to your text when inserting text in keypad mode or filling text in all three editing modes. Lines exceeding the current **SET SCREEN** width are always wrapped in line mode.

Example: Shows that SET NOTRUNCATE is currently in effect.

```
*SHOW TRUNCATE
notruncate
```

SHOW VERIFY Command
Line

Syntax:

```
SHOW VERIFY
```

Description:

The SHOW VERIFY command tells you whether **SET VERIFY** or SET NOVERIFY is currently in effect for your editing session. If SET VERIFY is in effect, EDT displays the commands in the startup command file or EDT macro that are being processed. SET NOVERIFY is the default.

Example: Shows that the default NOVERIFY is in effect.

```
*SHOW VERIFY
noverify
```

Syntax:

```
SHOW VERSION
```

Description:

The SHOW VERSION command displays the version of EDT that is being used by your operating system. The command also displays EDT's copyright notice. There is no way for you to reset the version of EDT currently available on your system from within EDT. (EDT has no SET VERSION command.) However, this information is useful when reporting EDT problems.

The version syntax is:

```
support version.update-edit patch
```

Support is a letter that indicates the support status of the version of EDT that you are using. The possible letters are:

```
B  =  benchmark configuration; no support
D  =  demonstration configuration; no support
S  =  special customer configuration; negotiated support
T  =  field test version; not supported after field test ends
V  =  released version; full support
X  =  experimental version; no support.
```

Version is the version number. If it does not match the number on the title page of this manual, the book does not apply to your version of EDT. **Update** represents an update to the base version. The update involves minor changes to the documentation. Check to be sure that this number is the same as that given in the front of this manual.

Edit represents a minor change to EDT that has no documentation impact. **Patch** is an optional letter, which represents a minor change to EDT with no documentation impact. The change has been applied by means of a binary patch.

When you fill out a Software Performance Report (SPR) for EDT, be sure to include the full version number that EDT displays when you issue the SHOW VERSION command. Use the space provided on the SPR form to enter the version number.

Example: Explains the meaning of the version number.

```
*SHOW VERSION
V3.00-1         COPYRIGHT (c) DIGITAL EQUIPMENT CORPORATION 1983
```

This is a fully supported released version. The version number is **3**. The **00** indicates that this version has no updates. The **1** in the edit position indicates a minor change that has no impact on the documentation. No patch letter is present.

SHOW WORD Command
Line

Syntax:

```
SHOW WORD
```

Description:

The SHOW WORD command tells you how EDT is interpreting word boundaries when you use the word entity in keypad or nokeypad mode. When NODELIMITER is in effect, EDT does not consider word delimiters such as line terminators and horizontal tabs to be words themselves. (The space word boundary is always considered as part of a word, never as a separate word.) The default is **SET WORD DELIMITER**.

Example: Shows that SET WORD NODELIMITER is currently in effect.

```
*SHOW WORD
nodelimiter
```

Syntax:

```
SHOW WRAP
```

Description:

The SHOW WRAP command tells you if a **SET WRAP** command is in effect and if so, what the SET WRAP value is. SET WRAP has two functions. It causes EDT to wrap lines when you are inserting text in keypad mode and it sets the right margin for all three EDT **FILL** functions. SET WRAP always overrides the current **SET SCREEN** width in determining the line length for the FILL functions.

In the screen modes, **SET NOTRUNCATE** wraps lines that are already in your buffer for display purposes only; it has no effect on the text.

Example 1: Shows that the default NOWRAP setting is in effect.

```
*SHOW WRAP
nowrap
```

Example 2: Shows that SET WRAP 50 is currently in effect.

```
*SHOW WRAP
50
```

SHR (Shift Right) Command
Nokeypad

Syntax:

```
[count]SHR
```

Description:

The SHR (shift right) command moves the entire current buffer eight character positions (one tab stop) to the right. SHR only works after you have used the **SHL** (shift left) command. When SHL has been used to move the buffer over one tab stop, SHR will move the buffer back to its original position.

You cannot shift text further to the right with SHR than the text has been moved to the left. Once column 1 of your buffer is again at the left edge of the screen, further SHR commands have no effect on the display.

No characters are actually added or deleted from the buffer when SHR or SHL is used. The effects of these commands are merely visual and temporary. On a VT100-type terminal with advanced video option, you can use the **SET SCREEN** command to display up to 132 characters on a line. However, the size of each character is reduced.

The count specifier designates the number of tab stops to shift right. When the count is 4, the buffer is shifted 32 character positions to the right. If the count used with the SHL command is 5, and you then give the command 4SHR, the buffer will end up still shifted eight character positions to the left. If you use the SHR command with a count specifier greater than the number of left shifts that have been made, EDT moves the text back to its original position and ignores the remaining repeats.

You can define the GOLD/→ key on VT100-type terminals to perform the SHR function in keypad mode. The following **DEFINE KEY** command can be included in your startup command file or issued during your editing session:

```
DEFINE KEY GOLD 14 AS "SHR."
```

Example: Uses the SHL command to move the line 16 columns to the left. Then uses SHR to restore the characters in columns 9 through 16 to view.

```
AAAAAAAbbbbbbbbCCCCCCCCddddddddFFFFFFFFggggggggHHHHHHHHiiiiiiiiJJJJJJJJKKKKKKK◇

2SHL

CCCCCCCddddddddFFFFFFFFggggggggHHHHHHHHiiiiiiiiJJJJJJJJKKKKKKKKlllllllMMMMMMM◇

SHR

bbbbbbbCCCCCCCCddddddddFFFFFFFFggggggggHHHHHHHHiiiiiiiiJJJJJJJJKKKKKKKKlllllll◇
```

Syntax:

```
[+|-][count]SN
```

Description:

The SN (substitute next) command is a substitute command that does not take any string specifiers. It uses strings that have been stored in the search buffer (string-1) and the substitute buffer (string-2). (Remember that you cannot access either of these buffers.) The SN command looks for the next occurrence of string-1 (the current search string) and replaces it with string-2 (the current substitute string).

Since SN uses the contents of the search buffer as its search string, you must be aware of the current contents of that buffer. If you have used a string entity after your initial S (substitute) command, that new string will now be in the search buffer, not the string you want to use with the SN command.

With the count specifier you can repeat the substitution several times. Use the sign specifier to change EDT's direction for the SN command.

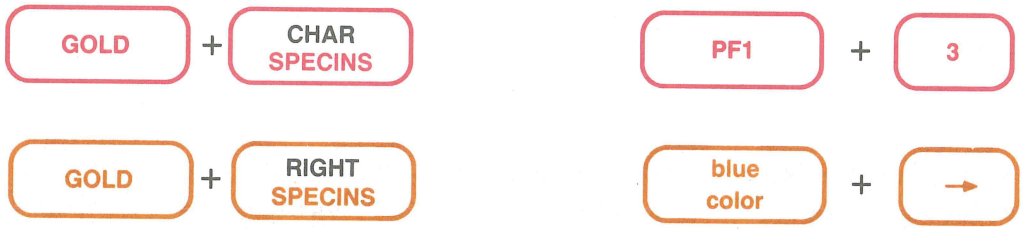
Example: First uses the S command to make the initial substitution on line 2. Then uses SN to make the substitution on line 5.

```
BASIC, a Beginner's All-purpose Symbolic Instruction Code,
is a languge that requires only an understanding of English.
BASIC was developed at Dartmouth College for use by
students who were unfamiliar with computers and needed a
languge related to everyday speech.

S/languge/language/

SN

BASIC, a Beginner's All-purpose Symbolic Instruction Code,
is a language that requires only an understanding of English.
BASIC was developed at Dartmouth College for use by
students who were unfamiliar with computers and needed a
language related to everyday speech.
```

Related Commands: Keypad – SUBS
Line – SUBSTITUTE NEXT

SPECINS (Special Insert) Function
Keypad

Keypad Designations:



Description:

SPECINS (special insert) enables you to insert any character from the DEC Multinational Character Set into your text using the character's decimal equivalent value. You can use SPECINS to enter ASCII control characters, such as CTRL/L, or letters with diacritical marks such as the umlaut (¨) or acute accent (´). **ASC.** is the nokeypad definition for SPECINS.

To use SPECINS, first press GOLD. Next, type the decimal equivalent number for the character you want to insert. Use the main keyboard digits to type this number; do not use the keypad number keys. EDT displays the number you typed at the bottom of the screen. You can use the DELETE key to edit the number. Now press GOLD again, this time followed by the SPECINS key. The EDT symbol for the character you inserted appears on the screen to the left of the cursor.

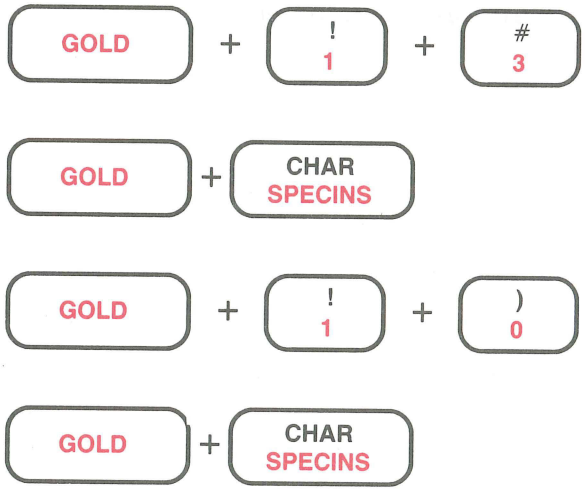
Each time you want to enter a special character, you must repeat the entire procedure. You cannot enter two characters with one SPECINS function, nor can you use the GOLD repeat feature to enter the same character several times in one location.

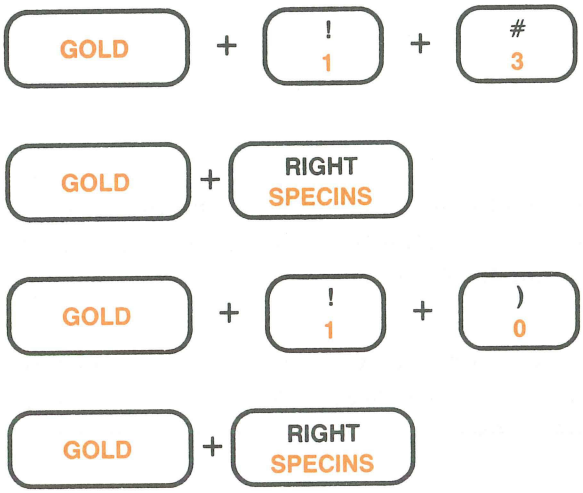
SPECINS cannot be used if **SET NOREPEAT** is in effect for your editing session.

The maximum decimal character value for SPECINS is 255. The DEC Multinational Character Set in Appendix G lists the decimal equivalent value for each character from 0 through 255.

Example: Inserts first a <CR>, then an <LF> near the end of the line.

```
    for the last time.  Then when you need to get another Shipment
```






```
for the last time.  Then when you need to get another <CR><LF>Shipment
```

Related Commands: Nokeypad – ASC, ^ (circumflex)

SSEL (Search and Select) Command
Nokeypad

Syntax:

```
[+|-]SSEL[+|-]"string"
```

Description:

The SSEL (search and select) command allows you to find a string and designate it as a select range in one operation. The string must be enclosed in quotation marks (single – ', or double – "). Do not use spaces to separate the command from the string. However, you can include leading spaces in the search string.

When you use SSEL, the contents of the search buffer are overwritten by the string. To search and select that string again (or whatever string is currently in the search buffer), simply type two quotation marks with no intervening characters, as follows:

```
SSEL""
```

You can use a select range with line mode commands by giving the line mode range specifier SELECT. However, line mode requires that a select range contain only whole lines. Thus, you can only use the SSEL command for this purpose if the string is an entire line of text. You might find it easier to create longer select ranges with the nokeypad **SEL** command.

Example: First creates a select range from the search string ⓈⓅ**EQUIPMENT CORPORATION**. Then deletes the select range. Finally creates a new select range from the next occurrence of the same search string.

```
DIGITAL EQUIPMENT CORPORATION
Digital Equipment Corporation is located in Maynard, MA.

SSEL"▵EQUIPMENT CORPORATION"

DSR

DIGITAL
Digital Equipment Corporation is located in Maynard, MA.

SSEL""

DIGITAL
Digital Equipment Corporation is located in Maynard, MA.
```

Syntax:

```
/STAY
```

Description:

The /STAY qualifier is used with the line mode **TYPE** command. All EDT qualifiers must be preceded by a slash. /STAY keeps the EDT pointer at its current position, regardless of which lines have just been displayed by the TYPE command.

Example 1: Shows that EDT remains on line 35 even after typing lines 2 through 4.

```
     35       whenever you are ready to send me the information,

*TYPE 2 THRU 4 /STAY

      2       August 17, 1984
      3
      4       Dr. Hugo Arndt

*TYPE .

     35       whenever you are ready to send me the information,
```

Example 2: Displays the contents of the PASTE buffer while still in the MAIN buffer.

```
*TYPE =PASTE /STAY

      1       whether or not you have made your decision.
      2       But, of course, we would welcome your joining
  [EOB]

*TYPE .

    284       the details of our current business plan.
```

string specifier
Keypad
Line
Nokeypad

Syntax:

```
"string"
'string'
/string-1/string-2/
```

Description:

The string specifier is generally used either to locate characters in a buffer or to replace the located characters. When a string specifier is used to locate a piece of text, it is referred to as the search string. All three editing modes use search strings. Generally a search string must be enclosed in quotation marks (single – ' or double – ") to distinguish it from other characters in the command line.

When strings are used in EDT substitute commands, they are generally referred to as string-1 or string-2. String-1 is always a search string. EDT must locate that string and then replace it with string-2, the substitute string. Strings in substitute commands must be surrounded by delimiters to separate them from the other characters in the command line. A variety of nonalphanumeric characters can be used as delimiters as long as the delimiter does not appear in either string-1 or string-2. (In line mode you cannot use the percent sign – % or the underscore – _ as string delimiters.) All three delimiters in a single substitute command must be identical.

Whenever you issue a search string, EDT overwrites the contents of the search buffer. Similarly, when you issue a substitute string, EDT overwrites the contents of the substitute buffer.

The string entity in nokeypad mode refers to all the characters in the text between the initial cursor position and the start of the search string. However, only the search string itself is stored in the search buffer.

The search and substitute buffers cannot be edited or entered. Their names never appear in the SHOW BUFFER list. You can use the nokeypad **CLSS** (clear search string) command to delete the contents of the search buffer.

EDT has a number of ways to perform searches. See the discussion of the **SET SEARCH** command in this chapter and in Chapter 6 for more information about the EDT search parameters.

Examples:

```
Search for:  lollypop
```

Moves the cursor to the string **lollypop**.

Moves the cursor to the next occurrence of the string **lollypop**.

```
*DELETE "afternoon"
```

Deletes the line containing the string **afternoon**.

```
*TYPE "January"
```

Displays the line containing the string **January**.

```
*SUBSTITUTE NEXT/line/keypad/
```

Finds the next occurrence of the string **line** and replaces it with the string **keypad**.

```
S/COBOL/BASIC/
```

Substitutes the string **BASIC** for the string **COBOL**.

```
D"."
```

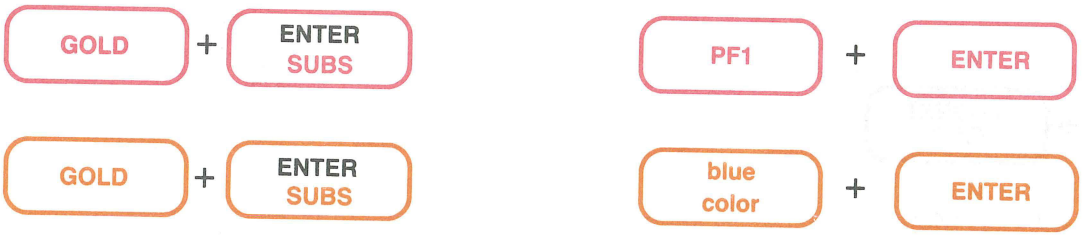
Deletes all characters starting at the current cursor position and continuing until the next period is reached.

```
SSEL"Friday, May 13th"
```

Finds the string **Friday, May 13th** and makes it a select range.

SUBS (Substitute) Function
Keypad

Keypad Designations:



Description:

SUBS (substitute) replaces the current search string with the contents of the PASTE buffer. In order to use SUBS you must first put the string you want to replace in the search buffer and the new text in the PASTE buffer. All searches and substitutions are made in the current direction.

(CUTSR = DELETE PASTEKS""). is the nokeypad definition for SUBS. This means that the select range – in most cases the current search string – is deleted from the current buffer and placed in a buffer named DELETE. The contents of the PASTE buffer are inserted in the text and the cursor is placed on the last character of the inserted text if EDT's direction is forward. (If EDT's direction is backward, the cursor is positioned on the first character of the inserted text.) Finally, EDT moves to the next occurrence of the current search string.

Using SUBS involves four steps:

1. Put the search string in the search buffer.
2. Put the replacement text in the PASTE buffer.
3. Locate the search string.
4. Press GOLD + SUBS.

1. The easiest way to load the search buffer is with the **FIND** function. You can also use any line mode or nokeypad command that involves a search string. Remember, the search buffer cannot be entered or edited.
2. There are two ways to load the PASTE buffer.
   - You can type the replacement text in your current buffer, make it a select range, and the use CUT to transfer it to the PASTE buffer.
   - Since you can enter the PASTE buffer and edit its contents, you can use the line mode **FIND** command to move to the PASTE buffer and then insert the replacement text there. Use the FIND command to return to the buffer you are editing.
3. If you reverse the order of steps 1 and 2, the cursor will already be at the search string. Otherwise, you must be sure that the cursor is positioned on the first character of the search string before you press SUBS. This is because SUBS preforms the substitution first and then moves to the next occurrence of the search string. The order allows you to decide whether you want to perform the substitution on that instance of the search string or go on to the next one. (Use **FNDNXT** to skip the substitution on the current search string match and advance to the next occurrence.)

4. The final step calls for pressing GOLD and then SUBS. If the cursor is not on the search string, EDT prints the message **No select range active**. If there is no other search string match in the remaining portion of your buffer, EDT prints the message **String was not found**.

SUBS is the only substitute function that can use a line terminator in the replacement text.

Example 1: First locates the string **languge** in line 2. Then puts the substitute string in the PASTE buffer. Replaces the mistake on line 2 and then the one on line 5.

```
BASIC, a Beginner's All-purpose Symbolic Instruction Code,
is a languge that requires only an understanding of English.
BASIC was developed at Dartmouth College for use by
students who were unfamiliar with computers and needed a
languge related to everyday speech.
```

GOLD + FNDNXT FIND

```
Search for:  languge
```

ENTER SUBS

(Now type the string **language** at the current cursor position.)

```
BASIC, a Beginner's All-purpose Symbolic Instruction Code,
is a languagelanguge that requires only an understanding of English.
BASIC was developed at Dartmouth College for use by
students who were unfamiliar with computers and needed a
languge related to everyday speech.
```

SELECT RESET + GOLD + 8 + ← + CUT PASTE

GOLD + ENTER SUBS + GOLD + ENTER SUBS

```
BASIC, a Beginner's All-purpose Symbolic Instruction Code,
is a language that requires only an understanding of English.
BASIC was developed at Dartmouth College for use by
students who were unfamiliar with computers and needed a
languagge related to everyday speech.
   .
   .
   .
String was not found
```

Example 2: Using the line terminator in both the search and replacement strings, inserts a dollar sign ($) at the beginning of each line.

```
29.99
35.95
17.00
```

GOLD + FNDNXT FIND

```
Search for:  ^M
```

(Press RETURN in response to Search for: to put the line terminator in the search buffer.)

ENTER SUBS

RETURN + $ 4

(You press RETURN and then $ to put both the terminator and the dollar sign characters in the PASTE buffer.)

SELECT RESET + ← + ← + CUT PASTE

(Move the cursor back to its original position above the first number.)

GOLD + ENTER SUBS + GOLD + ENTER SUBS + GOLD + ENTER SUBS

```
$29.99
$35.95
$17.00
```

Related Commands: Line – SUBSTITUTE NEXT
Nokeypad – SN (substitute next)

Syntax:

```
SUBSTITUTE/[string-1]/string-2/ [=buffer] [range][/BRIEF[:n]] [/QUERY]
     [/NOTYPE]
```

Description:

The SUBSTITUTE command replaces string-1 with string-2 everywhere in the specified range. If no range is given, the substitution is made on the first occurrence of the string in the current line. You must reissue the SUBSTITUTE command or use the **SUBSTITUTE NEXT** command to change subsequent occurrences.

String-1 and string-2 must be surrounded by delimiters. You can use any nonalphanumeric character as a delimiter (except the percent sign – % and the underscore – _) provided that the character does not appear in either string-1 or string-2. The three delimiters must be identical. Notice that the delimiters are not optional.

The SUBSTITUTE command displays the lines in which substitutions were made as well as a message indicating the number of substitutions. If EDT is unable to process the command, it prints the message **No substitutions**.

The SUBSTITUTE command can take several specifiers and qualifiers. The range specifier tells EDT to make as many substitutions as there are occurrences of string-1 in the range. If your range covers only one line, but there are two occurrences of string-1 in that line, both substitutions are made. Without the range specifier, only the first substitution would be made.

You can also perform substitutions in other buffers. If you do not use a range specifier following the buffer name, all occurrences of string-1 in the specified buffer are found and replaced with string-2. EDT leaves you in the specified buffer. You must use another command to return to your original buffer.

You do not have to include both string-1 and string-2 in your SUBSTITUTE command, but you must include at least one of them. If you omit string-1, EDT always uses the current search string. If you omit string-2, EDT always deletes string-1 and inserts nothing. If you use a string for range, EDT considers that to be the current search string if string-1 is omitted. The following samples show what happens:

```
This is file A.     SUBSTITUTE/file/buffer/     This is buffer A.
This is file B.     SUBSTITUTE///             Search string cannot be null
This is file C.     SUBSTITUTE/file//           This is  C.
This is file D.     SUBSTITUTE//buffer/         This is buffer D.
This is file E.     SUBSTITUTE//buffer/ "e."    This is file buffer *
```

*In this line, EDT first located the appropriate line using the search string **e.** to find it. The search buffer no longer contains the string **file**; the current search string is **e.** So, EDT replaces the string **e.** with the substitute string. Only use a string range specifier if you include string-1 in your SUBSTITUTE command or if you want to replace the string range specifier with the substitute string.

The /**BRIEF** qualifier limits the number of characters displayed for each line in which a substitution is made. The default value for the **n** specifier is 10. By using the **n** specifier, you can have more or fewer than 10 characters displayed. When the /BRIEF qualifier is not used, the entire line is printed, unless the /NOTYPE qualifier is in effect.

The /**QUERY** qualifier prints the line containing string-1 before the substitution is made. EDT then prompts you with the question mark (?) to let you decide whether to make the substitution in that line or go on to the next occurrence of string-1. Type one of these four responses to the question mark prompt: Y (YES), N (NO), A (ALL), or Q (QUIT).

When you use the /**NOTYPE** qualifier, EDT does not display the lines after making the substitutions. However, EDT does print the message that tells you how many substitutions were made.

Note: the slashes preceding the qualifiers /BRIEF, /QUERY, and /NOTYPE are signals to EDT, not delimiters. Only the slash can indicate that the following letters refer to a qualifier.

Example 1: Replaces the first occurrence of **this editor** with **EDT** on the current line.

```
     95        with this editor.  This editor can also

*SUBSTITUTE/this editor/EDT/

     95        with EDT.  This editor can also
 1 substitution
```

Example 2: Replaces both occurrences of **this editor** with **EDT** on the current line.

```
     95        with this editor.  This editor can also

*SUBSTITUTE/this editor/EDT/ 95

     95        with EDT.  EDT can also
 2 substitutions
```

(Note that **SUBSTITUTE/this editor/EDT/** . produces the same results.)

Example 3: Replaces all occurrences of **April 19** with **May 16** in lines 15 through 30.

```
     15        The meeting is scheduled for
     16        April 19, 1984 at 7:00 p.m. at the Campus Inn.
        .
        .
        .
     29        All programmers planning to attend the April 19th
     30        meeting should contact Marsha Lambert as soon as

*SUBSTITUTE/April 19/May 16/ 15 THRU 30

     16        May 16, 1984 at 7:00 p.m. at the Campus Inn.
     29        All programmers planning to attend the May 16th
 2 substitutions
```

Example 4: Replaces **2B** with **1C** for the entire buffer, but uses the /QUERY qualifier to let you decide individually which instances to replace. Note that the delimiter character is the exclamation point (!).

```
      1        John Hershey            2A
      2        Max Greenstein          2B
      3        Jennifer Grogan         2B
      4        Larry Sadler            2B
      5        Quincy Marcus           2A
      6        Shirley Green           2A
      7        Thomas Orlovsky         2B
      8        Theodore Rossmann       2B
      9        Marion Andrews          2B

*SUBSTITUTE!2B!1C! WHOLE /QUERY

      2        Max Greenstein          2B
?Y
      2        Max Greenstein          1C
      3        Jennifer Grogan         2B
?N
      4        Larry Sadler            2B
?Y
      4        Larry Sadler            1C
      7        Thomas Orlovsky         2B
?N
      8        Theodore Rossmann       2B
?A
      8        Theodore Rossmann       1C
      9        Marion Andrews          1C
4 substitutions
```

Example 5: Replaces **PICTURE** with **PIC** on lines 21 through 23. Only the number of substitutions made appears, not the altered lines. Uses the TYPE command to verify the change.

```
     21        00120    01  STUDENT-CARD
     22        00121        05  STUDENT-NAME      PICTURE IS A(20)
     23        00122        05  SOC-SEC-NO        PICTURE IS 9(9)

*SUBSTITUTE<PICTURE<PIC< 21 THRU 23 /NOTYPE
 2 substitutions

*TYPE 22 THRU 23

     22        00121        05  STUDENT-NAME      PIC IS A(20)
     23        00122        05  SOC-SEC-NO        PIC IS 9(9)
```

Example 6: Replaces the string **125 State** with the string **1001 Main**. Displays only the first 20 characters on the line after the substitution is made.

```
     15        125 State Street, North Adams, Massachusetts

*SUBSTITUTE'125 State'1001 Main' /BRIEF:20

     15        1001 Main Street, No
 1 substitution
```

Related Commands: Keypad – SUBS
Nokeypad – S (substitute)

Syntax:

```
[SUBSTITUTE] NEXT[/[string-1]/string-2/]
```

Description:

The SUBSTITUTE NEXT command causes EDT to search for the next occurrence of string-1 and replace it with string-2. The SUBSTITUTE NEXT command takes neither a buffer specifier nor a range specifier. It can perform only one substitution at a time. Hence EDT never prints the number of substitutions for the command. If EDT cannot find string-1, it stops at the *end* of the buffer. No message is printed.

Since the word SUBSTITUTE is optional in the SUBSTITUTE NEXT command, this command is sometimes referred to simply as the NEXT command. If you do include SUBSTITUTE or its abbreviation – S – in your command line, be sure to put a space before the word NEXT or its abbreviation – N.

The string-1 and string-2 specifiers must be surrounded by delimiters. You can use any non-alphanumeric character as a delimiter (except the percent sign – % and the underscore – _), provided that the character does not appear in either string-1 or string-2. The three delimiters must be identical.

The string-1 or string-2 specifier can sometimes be omitted from the SUBSTITUTE NEXT command. If you omit string-1, EDT always uses the current search string. If you omit string-2, but supply string-1, EDT deletes the search string from the text and replaces it with nothing. If you supply neither string-1 nor string-2 *and* do not include the delimiters, EDT uses the current search string for string-1 and the current substitute string for string-2. If you omit both strings, but include the delimiters, EDT displays an error message.

The following samples show what happens when you omit one or both string specifiers:

```
This is file A.     SUBSTITUTE NEXT/file/buffer/     This is buffer A.
This is file B.     SUBSTITUTE NEXT///        Search string cannot be null
This is file C.     SUBSTITUTE NEXT/file//           This is  C.
This is file D.     SUBSTITUTE NEXT//buffer/         This is buffer D.
This is file E.     SUBSTITUTE NEXT                  This is buffer E.
```

Example 1: Moves to line 6 to replace **formula** with **FORmula**.

```
     5       FORTRAN gets its name from the two words
     6       formula and translation.

*SUBSTITUTE NEXT/formula/FORmula/

     6       FORmula and translation.
```

Example 2: Uses SUBSTITUTE to replace **84** with **85** on line 12. Then uses SUBSTITUTE NEXT without any strings to make the substitution on line 13.

```
    12        January 1, 1984
    13        May 30, 1984

*SUBSTITUTE/84/85/ 12

    12        January 1, 1985
1 substitution

*SUBSTITUTE NEXT

    13        May 30, 1985
```

Example 3: First replaces **the VT52** with **the VT100** on line 59. Then using the same search string, replaces it with **a VT100**.

```
    58        EDT is optimized for use on a high-speed
    59        terminal with a keypad, such as the VT52.
    60        You can use the VT52 keypad both as a
    61        numerical keypad and as an editing keypad.

*SUBSTITUTE NEXT/the VT52/the VT100/

    59        terminal with a keypad, such as the VT100.

*SUBSTITUTE NEXT//a VT100/

    60        You can use a VT100 keypad both as a
```

Example 4: Replaces the string **input-output** with the string **I/O**. Uses the colon as the delimiter character.

```
    37        Computer terminals are input-output devices.

*SUBSTITUTE NEXT:input-output:I/O:

    37        Computer terminals are I/O devices.
```

Related Commands: Keypad – SUBS
Nokeypad – SN (substitute next)

Keypad Designations:

Description:

TAB (CTRL/I) moves text to the right. The number of column positions that the text moves depends on the cursor position, the **SET TAB** value, if one is in effect, and the indentation level count, if one is in effect. (SET NOTAB is the default.) **TAB.** is the nokeypad definition for TAB.

EDT has preset tab stops every eight characters, regardless of how your terminal is set. If no SET TAB command has been given, pressing TAB moves the cursor character, as well as all the characters on the current line to the right of the cursor, to the nearest preset tab position. Text is always moved to the right, regardless of EDT's current direction.

When a SET TAB value is in effect, TAB moves the entire line to the column designated by the SET TAB value only if the cursor is located in column one. If the cursor is located anywhere else on the line, TAB moves the text to the nearest preset EDT tab stop.

If a tab indentation level count is in effect and the cursor is located in column one of the line, TAB moves the text to the indentation level position. The indentation level count is determined by three functions: (1) **CTRL/A**, which can be used to compute the indentation level count, (2) **CTRL/D**, which decrements the count, and (3) **CTRL/E**, which increments the count. Use the **SHOW TAB** command to find out the current SET TAB value and the indentation level count.

CTRL/T indents whole lines of text by the SET TAB value. See Chapter 7 for more information on EDT's tabbing facility.

The TAB key and CTRL/I always have identical functions in EDT. When you redefine the TAB key, you redefine CTRL/I. To redefine the TAB key using the line mode **DEFINE KEY** command, type DEFINE KEY CONTROL I. When you want to find out the definition of the TAB key, type **SHOW KEY** CONTROL I.

Example: Using the default EDT tab settings (SET NOTAB in effect), indents the lines of text, each one eight columns further than the one above.

```
This is the first line of text.
This is the second line of text.
This is the third line of text.
This is the fourth line of text.
```

(Start with the cursor at the beginning of the second line.)

(Move the cursor to the beginning of the third line.)

(Move the cursor to the beginning of the fourth line.)

```
This is the first line of text.
        This is the second line of text.
                This is the third line of text.
                        This is the fourth line of text.
```

Related Commands: Nokeypad – TAB

Syntax:

```
[count]TAB
```

Description:

The TAB command moves text to the right, regardless of EDT's direction. The number of column positions that the text moves depends on the cursor position, the **SET TAB** value, if one is in effect, and the indentation level count, if one is in effect. (SET NOTAB is the default.)

EDT has preset tab stops every eight characters, regardless of how your terminal is set. If no SET TAB command has been given, the TAB command moves the cursor character, as well as all the characters on the current line to the right of the cursor, to the nearest preset tab position.

When a SET TAB command has been given, the TAB command moves the entire line to the column designated by the SET TAB value only if the cursor is located in column one. If the cursor is located anywhere else on the line, the TAB command moves the text to the nearest preset EDT tab stop.

If a tab indentation level count is in effect and the cursor is located in column one of the line, the TAB command moves the text to the indentation level position. The indentation level count is determined by three functions: (1) **TC** (tab compute) which can be used to compute the indentation level count, (2) **TD** (tab decrement) which decrements the count, and (3) **TI** (tab increment) which increments the count. Use the **SHOW TAB** command to find out the current SET TAB value and the indentation level count.

The **TADJ** command indents whole lines of text by the SET TAB value. See Chapter 7 for more information on EDT's tabbing facility.

Example: Using EDT's default tab settings (SET NOTAB), indents the lines of text, each one eight columns further than the one above.

```
This is the first line of text.
This is the second line of text.
This is the third line of text.
This is the fourth line of text.
```

(Start with the cursor at the beginning of the second line.)

```
TAB
```

(Move the cursor to the beginning of the third line.)

```
2TAB
```

(Move the cursor to the beginning of the fourth line.)

```
3TAB

     This is the first line of text.
             This is the second line of text.
                     This is the third line of text.
                             This is the fourth line of text.
```

Related Commands: Keypad – TAB, CTRL/I

Syntax:

```
TAB ADJUST [-]n [=buffer] [range]
```

Description:

The TAB ADJUST command enables you to indent whole lines to format outlines, indent computer programs, or create other types of layered text. In order for the command to work, you must establish a **SET TAB** value for your editing session. (The default is SET NOTAB.) The TAB ADJUST command cannot be used to arrange text in columns.

The **n** specifier is the multiple that EDT applies to the SET TAB value to determine how far to indent the line(s). For example, if your SET TAB value is 5, you must use an **n** specifier of 3 to indent a line 15 columns. The lines of text are always moved to the right unless you use a minus sign before the **n** specifier. The minus sign enables you to move indented text back toward the left margin. Notice that the **n** specifier is *not* optional. You must include a value of 1 to indent the text one tab stop.

When you specify a range, EDT indents every line specified by the range. If you include a buffer specifier with no range, the entire buffer is indented. When you omit both the buffer and range specifiers, EDT assumes that you have an active select range. Use the keypad **SELECT** function or the nokeypad **SEL** command to establish a select range. The select range can contain only whole lines.

To find out the current SET TAB value, use the **SHOW TAB** command. The level count displayed by the SHOW TAB command is not affected by the TAB ADJUST **n** specifier. The level count is only affected by the tab compute, tab decrement, and tab increment functions in the keypad and nokeypad modes.

To see the results of the TAB ADJUST command as you are working on the text, use the **SET NONUMBERS** command. With SET NUMBERS in effect, the EDT line numbers interfere with the indentation display.

Example: First establishes a SET TAB value of 3 and issues the SET NONUMBERS command. Then indents lines 2 through 5 so that each line moves one tab stop further over than the line above. Finally moves lines 4 and 5 to the left to change the format.

```
1       This is the first line.
2       This is the second line.
3       This is the third line.
4       This is the fourth line.
5       This is the fifth line.

*SET TAB 3
*SET NONUMBERS
*TAB ADJUST 1 2
*TAB ADJUST 2 3
*TAB ADJUST 3 4
*TAB ADJUST 4 5
*TYPE WHOLE

     This is the first line.
        This is the second line.
           This is the third line.
              This is the fourth line.
                 This is the fifth line.

*TAB ADJUST -2 4
*TAB ADJUST -4 5
*TYPE WHOLE

     This is the first line.
        This is the second line.
           This is the third line.
        This is the fourth line.
     This is the fifth line.
```

Related Commands: Keypad – CTRL/T
Nokeypad – TADJ (tab adjust)

Syntax:

```
[+|-][level-count]TADJ[+|-][entity-count]entity
```

Description:

The TADJ (tab adjust) command uses the value estabished by the line mode **SET TAB** command to indent lines of text. If no SET TAB value exists for the editing session, TADJ does nothing.

The level count specifier determines how many tab stops the text will be indented. The TADJ command moves text to the right. If you precede the level count specifier with a minus sign, you can move text, which has already been indented, back toward the left margin.

The entity count determines how many lines, paragraphs, or pages will be affected by the TADJ command. A minus sign before the entity or entity count determines whether EDT will work forward or backward in determining which entities to indent. Note that 5TADJL is *not* identical to TADJ5L. You can use both a level count and an entity count with TADJ, for example, 2TADJ4L.

To find out the current SET TAB value, use the line mode **SHOW TAB** command. EDT's default is SET NOTAB. Notice that the level count specifier has no effect on the tab level displayed by the SHOW TAB command.

Example: Using a SET TAB value of 3, indents line 2 one tab stop and line 3 two tab stops.

```
    This is the first line of text.
    This is the second line of text.
    This is the third line of text.

EXT SET TAB 3

NL TADJL

NL 2TADJL

    This is the first line of text.
       This is the second line of text.
          This is the third line of text.
```

Related Commands: Keypad – CTRL/T
Line – TAB ADJUST

TC (Tab Compute) Command
Nokeypad

Syntax:

```
TC
```

Description:

The TC (tab compute) command sets the indentation level count for the **TAB** command. The indentation level is the number of columns starting at the left of the screen that you want to leave blank before beginning a line of text. The indentation level in EDT must be a multiple of the **SET TAB** value. The indentation level count is the quotient of the indentation level divided by the SET TAB value. Use the line mode SET TAB command to establish the SET TAB value.

To use the TC command, first move the cursor as many positions to the right as you want to indent a block of text when you give the TAB command. Remember that this cursor position must be a multiple of the SET TAB value. If it is not, EDT prints an error message when you issue the TC command.

Once the cursor is in place, issue the TC command. You have now set the indentation level. If you divide the indentation level by the SET TAB value, you have the indentation level count. EDT retains the indentation level count for the remainder of the editing session or until you reset the count with a subsequent TC command, TD (tab decrement) command, or TI (tab increment) command. The TC command has no effect on EDT if SET NOTAB (the default) is in effect.

When you give the TC command, nothing happens to the text you are editing. This command simply tells EDT how to process the next TAB command(s). In order to indent any lines of text, you must issue the TAB command.

To find out the current SET TAB value and indentation level count, use the line mode **SHOW TAB** command.

Example: Using a SET TAB value of 3, establishes a tab level count of 5 in order to indent the first line of text 15 columns.

```
This is the first line of text.
This is the second line of text.

EXT SET TAB 3
```

(Move the cursor over 15 places so that it is on the **s** in **first**.)

```
TC

EXT SHOW TAB
tab size 3; tab level 5
```

(Now move the cursor back to the beginning of the first line.)

```
TAB

                    This is the first line of text.
     This is the second line of text.
```

Related Commands: Keypad – CTRL/A

TD (Tab Decrement) Command
Nokeypad

Syntax:

```
[count]TD
```

Description:

The TD (tab decrement) command decreases the current indentation level count. The indentation level count is generally set by the **TC** (tab compute) command. It can be altered by a subsequent TC command, TD command, or **TI** (tab increment) command. The TD command only resets the indentation level count. You must use the **TAB** command to move the text. TD has no effect on EDT if SET NOTAB (the default) is in effect. Use the line mode **SET TAB** command to establish a SET TAB value.

The count specifier determines the amount that the indentation level count decreases. If no specifier is given, the indentation level count decreases by one.

To find out the current SET TAB value and indentation level count, use the line mode **SHOW TAB** command.

Example: Using a SET TAB value of 4, sets the tab level count to 4. Then formats the text using TAB to indent the lines, and TD to change the indentation level.

```
This is the first line of text.
This is the second line of text.
This is the third line of text.
This is the fourth line of text.

EXT SET TAB 4

(Move the cursor 16 places to the right.)

TC

EXT SHOW TAB
tab size 4; tab level 4

(Move the cursor to the beginning of the first line.)

TAB

(Move the cursor to the beginning of the second line.)

TD

TAB
```

(Move the cursor to the beginning of the third line.)

```
TD

TAB
```

(Move the cursor to the beginning of the fourth line.)

```
TD

TAB
```

```
                    This is the first line of text.
                This is the second line of text.
            This is the third line of text.
        This is the fourth line of text.
```

```
EXT SHOW TAB
tab size 4; tab level 1
```

Related Commands: Keypad – CTRL/D

TGSEL (Toggle Select) Command
Nokeypad

Syntax:

Description:

The TGSEL (toggle select) command combines the functions of the **SEL** and **DESEL** commands into one. When there is an active select range, the TGSEL command cancels it, performing the same function as the DESEL command. When there is no active select range, TGSEL initiates the process of creating a select range, just as the SEL command does. The character that the cursor is on when you issue the TGSEL command constitutes one end of the select range. Move the cursor to the other end to define the limits of the select range.

You can use a select range with line mode commands by giving the line mode range specifier SELECT. However, line mode requires that the select range contain only whole lines.

Example: First creates a select range of lines 2 through 4 and then cancels it.

```
     Market Requirements
     Preliminary Business Plan
     Engineering Plan
     Training Plan
TGSEL

ƎL

     Market Requirements
     Preliminary Business Plan
     Engineering Plan
     Training Plan

TGSEL

     Market Requirements
     Preliminary Business Plan
     Engineering Plan
     Training Plan
```

Related Commands: Keypad – SELECT/RESET

Syntax:

```
[count]TI
```

Description:

The TI (tab increment) command increases the current indentation level count. The indentation level count is generally set by the **TC** (tab compute) command. It can be altered by a subsequent TC command, TI command, or **TD** (tab decrement) command. The TI command only resets the indentation level count. You must use the **TAB** command to move the text. TI has no effect on EDT if SET NOTAB (the default) is in effect. Use the line mode **SET TAB** command to establish the SET TAB value.

The count specifier determines the amount that the indentation level count increases. If no specifier is given, the indentation level count increases by one.

To find out the current SET TAB value and indentation level count, use the line mode **SHOW TAB** command.

Example: Using a SET TAB value of 5, sets the tab level count to 2. Then formats the text using TAB to indent the lines, and TI to change the indentation level.

```
This is the first line of text.
This is the second line of text.
This is the third line of text.
This is the fourth line of text.

EXT SET TAB 5

(Move the cursor 10 places to the right.)

TC

EXT SHOW TAB
tab size 5; tab level 2

(Move the cursor to the beginning of the first line.)

TAB

(Move the cursor to the beginning of the second line.)

TI

TAB
```

(Move the cursor to the beginning of the third line.)

```
TI

TAB
```

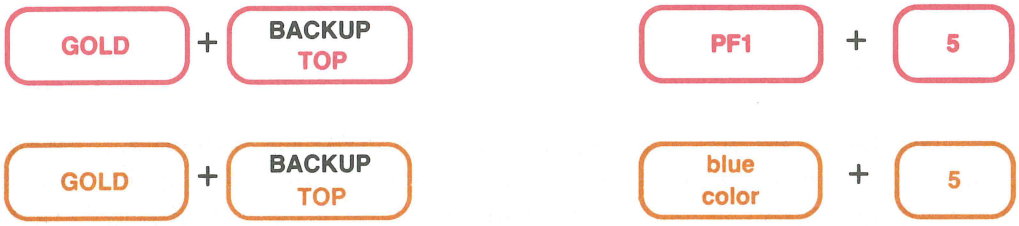
(Move the cursor to the beginning of the fourth line.)

```
TI

TAB
```

```
                This is the first line of text.
                    This is the second line of text.
                        This is the third line of text.
                            This is the fourth line of text.

EXT SHOW TAB
tab size 5; tab level 5
```

Related Commands: Keypad – CTRL/E

Keypad Designations:

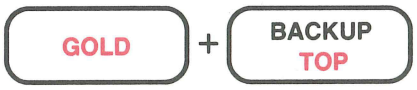


Description:

TOP moves the cursor to the first character position at the beginning of the buffer. TOP has no effect on EDT's current direction. **BR.** is the nokeypad definition for TOP.

Example: Moves the cursor from the middle of line 3 to the first character position in the buffer.

```
DATE:   June 16, 1985
TO:     Tom Langston
FROM:   Judy Tortini
  .
  .
  .
```




```
DATE:   June 16, 1985
TO:     Tom Langston
FROM:   Judy Tortini
  .
  .
  .
```

Related Commands: Nokeypad – BR

TOP Command
Nokeypad

Syntax:

```
TOP
```

Description:

The TOP command moves the line on which the cursor is located to the top of the display screen. If there are not enough lines between the current cursor position and the end of the buffer to fill the screen, TOP does nothing. When you use the TOP command, the cursor remains in the same position on the current line that it was in before TOP was issued.

Normally you must have 21 lines below the current line in order for TOP to have any effect on your screen display. However, if you have used the line mode **SET LINES** command to limit the number of lines per screen, TOP needs fewer lines between the cursor line and the end of the buffer in order to have an effect on the screen display.

Once a line has been placed at the top of the screen by the TOP command, EDT tries to keep it there as long as possible.

Example: Moves the line that the cursor is on to the top of the screen. SET LINES 5 limits the number of lines EDT displays.

```
EXT SET LINES 5

     This is the first line of text.
     This is the second line of text.
     This is the third line of text.
     This is the fourth line of text.
     This is the fifth line of text.

TOP

     This is the fourth line of text.
     This is the fifth line of text.
     This is the sixth line of text.
     This is the seventh line of text.
     This is the eighth line of text.
```

Related Commands: Keypad – PAGE

Syntax:

```
TYPE [=buffer] [range] [/BRIEF[:n]][/STAY]
```

Description:

The TYPE command is used to display lines of text at your terminal. If you omit both the buffer and range specifiers, EDT prints the current line. You can use the range specifier to print one or more lines anywhere in the current or specified buffer. If you use the buffer specifier without a range specifier, EDT moves to the named buffer and prints all its lines. (Use CTRL/C if you want to stop EDT from printing all the lines in a long buffer.) When you include the buffer specifier, EDT moves to the named buffer and remains there unless you use the /STAY qualifier.

The /**BRIEF** qualifier signals EDT to display only part of each line. If you omit the **n** specifier, EDT prints the first 10 characters (the default). Use the **n** specifier to have EDT print more or less than 10 characters per line.

The /**STAY** qualifier instructs EDT to leave its pointer at the current line, rather than relocate it at the first line of the buffer or range being displayed. Even if the lines being printed are in another buffer, the /STAY qualifier maintains EDT's position as it was before you issued the TYPE command. Thus, you can display the contents of another buffer without having EDT transfer you to that buffer.

The line mode <**null**> command is similar to TYPE, but it has no command word and cannot take any qualifiers.

Example 1: Displays the current line.

```
5       All classes begin at 8:30 a.m., unless otherwise
6       noted.  Please be prompt.  Course material has
7       been designed so that it fills the entire scheduled
8       time.  Breakfast will be provided, so no need to
9       stop for a cup of coffee on the way to class.

*TYPE

5       All classes begin at 8:30 a.m., unless otherwise
```

Example 2: Using the same text as in Example 1, displays lines 6 through 8.

```
*TYPE 6 THRU 8

6       noted.  Please be prompt.  Course material has
7       been designed so that it fills the entire scheduled
8       time.  Breakfast will be provided, so no need to
```

Example 3: Using the same text as in Example 1, displays only the first five characters of line 9.

```
*TYPE 9 /BRIEF:5

     9       stop
```

Example 4: Using the same text as in Example 1, displays line 5, but the current line does not change.

```
*TYPE 5 /STAY

     5       All classes begin at 8:30 a.m., unless otherwise
*TYPE .

     9       stop for a cup of coffee on the way to class.
```

Example 5: Moves to the PASTE buffer and displays every line.

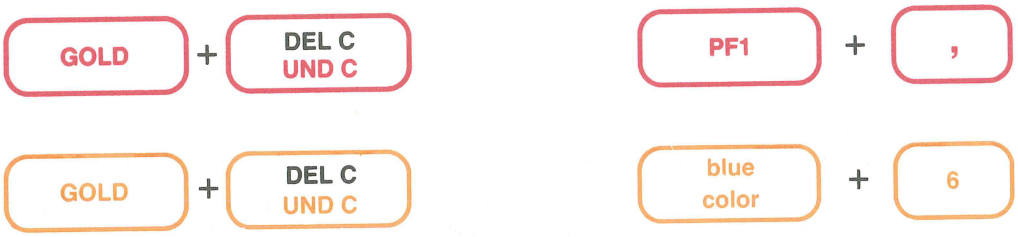
```
*TYPE =PASTE

     1       for the meeting.  When you are ready to put
     2       together an agenda, contact the Sales Manager
     [EOB]
```

Example 6: Returns to the MAIN buffer and displays lines 5 through 9.

```
*TYPE =MAIN 5 THRU 9

     5       All classes begin at 8:30 a.m., unless otherwise
     6       noted.  Please be prompt.  Course material has
     7       been designed so that it fills the entire scheduled
     8       time.  Breakfast will be provided, so no need to
     9       stop for a cup of coffee on the way to class.
```

Related Commands: Nokeypad – "move"

Keypad Designations:



Description:

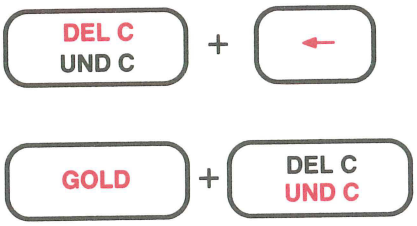
UND C (undelete character) inserts the current s of the delete character buffer into your text to the left ofthe cursor. The cursor character, as well as the text to the right of the cursor, ight. The cursor is now located on the inserted character if you used DEL C to delete the character. If you used DELETE to delete the character, the cursor is located to the right of the inserted character. **UNDC.** is the nokeypad definition for UND C.

The keypad functions **DEL C** and **DELETE** both place the character they delete in the delete character buffer. Each time you use DEL C or DELETE, the contents of the delete character buffer are overwritten. The buffer contains only the most recently deleted character. When you use a repeat count with DEL C or DELETE, only the last character deleted is in the delete character buffer. If no character has been deleted thus far during the current EDT session, UND C inserts nothing. Note that if you use the DELETE key to delete characters in a command line or prompt line, these characters are *not* stored in the delete character buffer and will not affect the character inserted by UND C.

EDT represents a line terminator as the character <CR> (CTRL/M, decimal 13) in all three of its delete entity buffers. Suppose you have a <CR> character in your text and you delete it. When you undelete this character, EDT changes the <CR> character into a line terminator and inserts the line terminator in your text, not the <CR> symbol.

Example: Using DEL C and UND C, reverses the order of the misplaced letters in **poeple** to **people**.

```
Many poeple have been aware of this problem for some time.
```




```
Many people have been aware of this problem for some time.
```

Related Commands: Nokeypad – UNDC (undelete character)

UNDC (Undelete Character) Command
Nokeypad

Syntax:

```
[count]UNDC
```

Description:

The UNDC (undelete character) command inserts the current contents of the delete character buffer into your text to the left of the cursor. The cursor character, as well as the text to the right of the cursor, moves to the right. The cursor is positioned on the inserted character if you used DC to delete the character. The cursor is positioned to the right of the inserted character if you used –DC (or D–C) to delete the character.

When you include a count specifier, EDT inserts the contents of the delete character buffer as many times as the count value.

Both the **DC** command and the **DELETE** key place the character they delete in the delete character buffer. Each time you use the DC command or the DELETE key, the contents of the delete character buffer are overwritten. The buffer contains only the most recently deleted character. When you use a count specifier with the DC command or with the DELETE key, only the last character deleted is in the delete character buffer. If no character has been deleted thus far during the current EDT session, UNDC inserts nothing. Note that if you use the DELETE key to delete characters in a command line, these characters are *not* stored in the delete character buffer and will not affect the character inserted by UNDC.

EDT represents a line terminator as the character <CR> (CTRL/M, decimal 13) in all three of its delete entity buffers. Suppose you have a <CR> character in your text and you delete it. When you undelete this character, EDT changes the <CR> character into a line terminator and then inserts the line terminator in your text, not the <CR> symbol.

Example: First loads the asterisk character (*) into the delete character buffer. Then inserts five asterisks in the text.

```
This is the end.
*
```

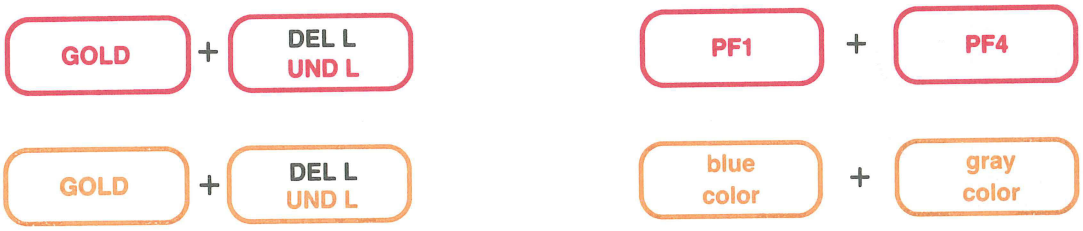
```
D-C
```

```
5UNDC
```

```
This is the end.
*****
```

Related Commands: Keypad – UND C

Keypad Designations:



Description:

UND L (undelete line) inserts the current contents of the delete line buffer to the left of the cursor. The cursor character, as well as the text to the right of the cursor, moves to a new line below the current line, if the buffer contents ends with a line terminator. Otherwise, text just moves to the right. The cursor is now located on the first character of the inserted text if you used DEL L or DEL EOL to delete the text. If you used CTRL/U to delete the text, the cursor is located to the right of the inserted text. **UNDL.** is the nokeypad definition for UND L.

The delete line buffer is loaded by using **DEL L**, **DEL EOL**, or **CTRL/U**. Each time one of these three functions is used, the contents of the delete line buffer are overwritten. The current contents of the buffer are the most recently deleted line or line portion. When you use a repeat count with a delete line function, only the last line or line portion that was deleted is in the delete line buffer. If no line has been deleted thus far in your EDT session, UND L T NOTHING.

EDT represents a line terminator as the character <CR> (CTRL/M, decimal 13) in all three of its delete entity buffers. Suppose you have a <CR> character in the text you are deleting. When you undelete this text, EDT changes the <CR> character into a line terminator and inserts the line terminator in the current buffer, not the <CR> symbol.

Example 1: Using DEL L and UND L, reorganizes the list so that the cities are in alphabetical order by state.

```
Montgomery, Alabama
Juneau, Alaska
Little Rock, Arkansas
Phoenix, Arizona
```

(Move the cursor to the **L** in **Little Rock**.)

GOLD + DEL L / UND L

```
Montgomery, Alabama
Juneau, Alaska
Phoenix, Arizona
Little Rock, Arkansas
```

Example 2: Using DEL L and UND L, inserts the second line of text after **you**.

```
When you give it more thought, please contact me.
have had a chance to
```

DEL L / UND L

(Move the cursor to the **g** in **give**.)

GOLD + DEL L / UND L

```
When you have had a chance to
give it more thought, please contact me.
```

Example 3: Uses DEL EOL and UND L to insert several words in the middle of some lines. Note that a leading space is included in the text stored in the delete line buffer.

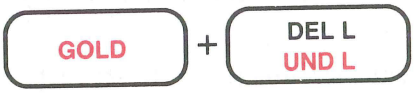
```
the company.  (Digital Equipment Corporation)
   .
   .
   .
After you have received the proper notification
from the company, you can
```

GOLD + EOL / DEL EOL

(Move the cursor back to the period [.] after the **y** in **company**.)

```
the company(Digital Equipment Corporation).
   .
   .
   .
After you have received the proper notification
from the company, you can
```

(Now move the cursor to the comma between **company** and **you**.)




```
the company (Digital Equipment Corporation).
   .
   .
   .
After you have received the proper notification
from the company(Digital Equipment Corporation), you can
```

Related Commands: Nokeypad – UNDL (undelete line)

UNDL (Undelete Line) Command
Nokeypad

Syntax:

```
[count]UNDL
```

Description:

The UNDL command inserts the current contents of the delete line buffer to the left of the cursor. The cursor character, as well as the text to the right of the cursor, moves to a new line below the current line, if there is a line terminator at the end of the buffer contents. Otherwise, the text just moves to the right. The cursor is now on the first character of the inserted test if you deleted the text in the forward direction (for example, DL, DNL, or DEL). If you deleted the text toward the beginning of the buffer (for example, DBL, –DEL, or –DL), the cursor is located to the right of the inserted text.

When the count specifier is used, the contents of the delete line buffer are inserted as many times as the count value.

The delete line buffer is loaded by using the **D** (delete) command with a line entity: L, BL, EL, or NL. Each time a DL type command is used, the contents of the delete line buffer are overwritten. The delete line buffer can only contain one line or line portion at a time. If you use a count specifier with the D command, for example D2L or 3DL, only the last line or line portion deleted is in the delete line buffer. If no line has been deleted thus far in your EDT session, UNDL inserts nothing.

EDT represents a line terminator as the character <CR> (CTRL/M, decimal 13) in all three of its delete entity buffers. Suppose you have a <CR> character in the text you are deleting. When you undelete this text, EDT changes the <CR> character into a line terminator and inserts the line terminator in the current buffer, not the <CR> symbol.

Example 1: Using DL and UNDL, reorganizes the list so that the cities are in alphabetical order by state.

```
Montgomery, Alabama
Juneau, Alaska
Little Rock, Arkansas
Phoenix, Arizona

DL
```

(Move the cursor to the **L** in **Little Rock**.)

```
UNDL

Montgomery, Alabama
Juneau, Alaska
Phoenix, Arizona
Little Rock, Arkansas
```

Example 2: Using DL and UNDL, inserts the second line of text after **you**.

```
When you give it more thought, please contact me.
have had a chance to

DL
```

(Move the cursor to the **g** in **give**.)

```
UNDL

When you have had a chance to
give it more thought, please contact me.
```

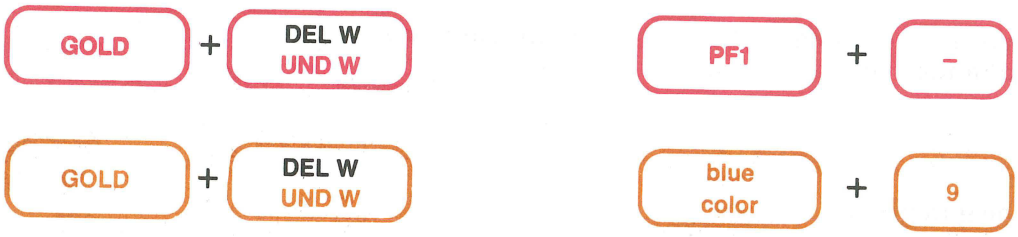
Example 3: Using DL and UNDL, duplicates the separator line in the current location.

```
This is the end.
*********

DL

2UNDL

This is the end.
*********
*********
```

Related Commands: Keypad – UND L

UND W (Undelete Word) Function
Keypad

Keypad Designations:



Description:

UND W (undelete word) inserts the current contents of the delete word buffer to the left of the cursor. The cursor character, as well as the text to the right of the cursor, moves to the right. The cursor is now located on the first character of the inserted word or word portion if you used DEL W to make the deletion. If you used LINEFEED, the cursor is located to the right of the inserted word or word portion. **UNDW.** is the nokeypad definition for UND W.

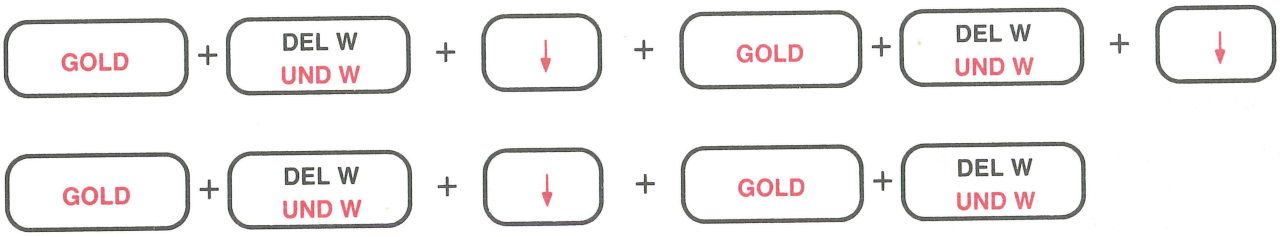
The delete word buffer is loaded by using **DEL W** or **LINEFEED** (CTRL/J, F13 – LK201). Each time you use DEL W or LINEFEED, the contents of the delete word buffer are overwritten. The current contents of the buffer are the most recently deleted word or word portion. When you use a repeat count with a delete word function, only the last word or word portion deleted is in the delete word buffer. If no word has been deleted thus far in your EDT session, UND W inserts nothing.

EDT represents a line terminator as the character <CR> (CTRL/M, decimal 13) in all three of its delete entity buffers. Suppose you have a <CR> character in the text you are deleting. When you undelete this text, EDT changes the <CR> character into a line terminator and inserts the line terminator in the current buffer, not the <CR> symbol.

Example: Using DEL W and UND W, inserts the word **PRINT** at the beginning of lines 1, 2, and 3.

```
LETTER1.MEM
LETTER2.MEM
LETTER3.MEM
PRINT LETTER4.MEM
```

(Move the cursor to the **L** in **LETTER1.MEM**.)




```
PRINT LETTER1.MEM
PRINT LETTER2.MEM
PRINT LETTER3.MEM
PRINT LETTER4.MEM
```

Related Commands: Nokeypad – UNDW (undelete word)

UNDW (Undelete Word) Command
Nokeypad

Syntax:

```
[count]UNDW
```

Description:

The UNDW (undelete word) command inserts the current contents of the delete word buffer to the left of the cursor. The cursor character, as well as the text to the right of the cursor, moves to the right. The cursor is now located on the first character of the inserted word or word portion if the deletion was made in the forward direction (for example, with DW or DEW). If the deletion was made toward the beginning of the buffer (for example, DBW or –DW), the cursor is located to the right of the inserted text.

When you include the count specifier, the contents of the delete word buffer are inserted as many times as the count value.

The delete word buffer is loaded by using the **D** (delete) command with a word entity: W, BW, or EW. Each time you use one of these commands, the contents of the delete word buffer are overwritten. The current contents of the buffer are the most recently deleted word or word portion. When you use a count specifier with a DW type command, only the last word or word portion is in the delete word buffer. If no word has been deleted thus far in your EDT session, UNDW inserts nothing.

EDT represents a line terminator as the character <CR> (CTRL/M, decimal 13) in all three of its delete entity buffers. Suppose you have a <CR> character in the text you are deleting. When you undelete this text, EDT changes the <CR> character into a line terminator and inserts the line terminator in the current buffer, not the <CR> symbol.

Example 1: Using DW and UNDW, inserts the word **PRINT** at the beginning of lines 1, 2, and 3.

```
LETTER1.MEM
LETTER2.MEM
LETTER3.MEM
PRINT LETTER4.MEM
```

```
DW
```

(Move the cursor to the **L** in **LETTER1.MEM**.)

```
UNDW
```

(Move the cursor to the **L** in **LETTER2.MEM**.)

```
UNDW
```

(Move the cursor to the **L** in **LETTER3.MEM**.)

```
UNDW
```

(Move the cursor to the **L** in **LETTER4.MEM**.)

```
UNDW
     PRINT LETTER1.MEM
     PRINT LETTER2.MEM
     PRINT LETTER3.MEM
     PRINT LETTER4.MEM
```

Example 2: Using DW and UNDW, creates a separator line below the current line.

```
     **** This is the end.

DW
```

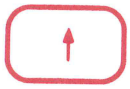
(Move the cursor to the beginning of the next line, which is blank.)

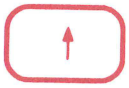
```
SUNDW
     This is the end.
     **** ***** ***** ***** *****
```

Related Commands: Keypad – UND W

Up Arrow
Keypad
Nokeypad

Key Designations:





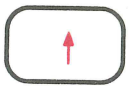
Description:

The Up Arrow key moves the cursor up one line toward the top of the buffer regardless of EDT's direction. **–V.** is the nokeypad definition for Up Arrow.

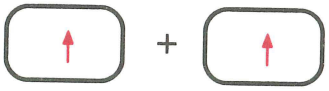
When you use the Up Arrow, EDT attempts to maintain the same vertical column as it moves the cursor from one line to the next. If there are not enough characters to fill out a line of text, the cursor moves to the end of the short line. If you continue to use Up Arrow, the cursor will return to the same vertical column for all lines that have enough characters. However, once you press some other key, EDT cancels the column position for Up Arrow and resets it the next time you use the function.

Example: Moves the cursor from the end of the last line to the end of the first line.

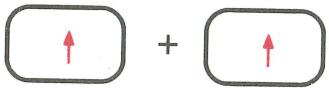
```
La Paz, Bolivia
Lima, Peru
Montevideo, Uruguay
Parimaribo, Surinam
Quito, Ecuador
Santiago, Chile
```



```
Quito, Ecuador
Santiago, Chile
```



```
Montevideo, Uruguay
Parimaribo, Surinam
Quito, Ecuador
```



```
La Paz, Bolivia
Lima, Peru
Montevideo, Uruguay
```

Keypad Designations:

Description:

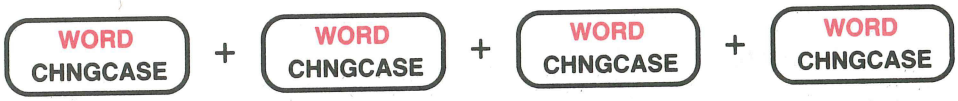
WORD moves the cursor to the beginning of the next word in the current direction (forward or backward, depending on whether **ADVANCE** or **BACKUP** is in effect). **W.** is the nokeypad definition for WORD.

An EDT word is any group of characters bounded by a space, horizontal tab, line feed (<LF>), vertical tab (<VT>), form feed (<FF>), or carriage return (<CR>). You can establish different word boundaries with the line mode **SET ENTITY** WORD command. Use the line mode **SHOW ENTITY** WORD command to find out the current boundary markers for the word entity.

The **SET WORD [NO]DELIMITER** command affects how EDT interprets the word boundaries. With SET WORD DELIMITER (the default) in effect, EDT considers all word boundaries, except the space, as words themselves.

Example: Moves the cursor four words to the right.

```
Mo e and more businesses are using computers.
```




```
More and more businesses are using computers.
```

Related Commands: Nokeypad – W (word)

WRITE Command
Line

Syntax:

```
WRITE file-specification [=buffer] [range]
          [/SEQUENCE[:initial[:increment]]]
```

Description:

The WRITE command copies text from an EDT buffer to an external file. The WRITE command has no effect on the buffer you are editing or the contents of any buffer in your EDT session. When you omit both the buffer and range specifiers from your WRITE command, EDT copies the entire contents of the current buffer to the external file. If you use the buffer specifier with no range, EDT copies the entire contents of the named buffer. When you issue a WRITE command with a buffer specifier, EDT does *not* move to the named buffer. You are still in the same buffer from which you issued the WRITE command.

The /**SEQUENCE** qualifier instructs EDT to put sequence numbers in the external file for use in subsequent editing sessions. The sequence numbers used with the WRITE command are not part of the text of the external file (as is the case with the **PRINT** command). However, these sequence numbers become the EDT line numbers when the file is edited with EDT. (The sequence numbers are ignored if you use the **INCLUDE** command to add the file to an EDT session already in progress.) Use the **RESEQUENCE** command to renumber your text when you are working in EDT. For more information on the use of the /SEQUENCE qualifier with the WRITE command, see the section on EDT line numbers in Chapter 7.

The WRITE command assigns EDT's default file attributes to the file it creates. Information on EDT's default file attributes appears in the system appendices.

Examples:

```
*WRITE DATA.DAT
```
Copies the entire current buffer to the file DATA.DAT.

```
*WRITE LETTER1.RNO 9 THRU 87
```
Copies lines 9 through 87 in the current buffer to the file LETTER1.RNO.

```
*WRITE MEMO11125.MEM =REVED11125
```
Copies the entire buffer REVED11125 to the file MEMO11125.RNO.

```
*WRITE YESNO.BAS =YESNOFIXD 123 THRU END
```
Copies lines 123 through the end in the buffer YESNOFIXD to the file YESNO.BAS.

```
*WRITE OVER.DAT /SEQUENCE:10:5
```
Copies the entire current buffer to the file OVER.DAT. Assigns sequence numbers to the external file. The numbers begin with 10 and increment by 5.

Syntax:

```
XLATEstring^Z
```

Description:

The XLATE command can be used only when EDT has been called by a running program on a VAX/VMS system. If you use it under other circumstances, EDT responds with an error message.

The string specifier enables you to pass information back to the calling program. CTRL/Z signals the end of the string. The nature of the string is determined by the type of program being run and the kind of information you need to pass to the program. For example, the information could be a DCL command or a variable that the program needs to continue running.

Since XLATE is available only as a nokeypad command, you might want to define a key to have the XLATE function. You can either include the exact information you want to pass to the program as the string specifier, or you can include a prompt so that you can enter the information when you press the function key. For example:

```
DEFINE KEY GOLD X AS "XLATEdirectory^Z."

DEFINE KEY GOLD X AS "XLATE?'Enter data: '^Z."
```

For more information on calling EDT from a running program and using the XLATE command, see Appendix D.

Appendix A
EDT Messages

EDT has a number of messages that it displays during the course of your editing session. Most of these messages are known as error messages because they indicate that EDT is unable to process the command you have issued. Other messages ask you to supply data so that EDT can perform a particular function. The "Working" message indicates that EDT is on the job, but needs a few seconds to process your command.

Some error messages use the circumflex (^) to point to the error in your command statement. This pointer indicates which portion of the command line contains the problem.

This appendix contains an alphabetical list of all the EDT messages with an explanation of each.

'.' required

The command required a period (.) or whatever other character is enclosed in the quotation marks.

Aborted by CTRL/C

This message appears whenever you use CTRL/C to cancel a command that is being processed.

Advance past bottom of buffer

Whenever you attempt to move the cursor past the end of a buffer ([EOB]), EDT displays this warning.

'AS' required

You are using the DEFINE KEY command, but you omitted the word **AS** from the syntax. For example, DEFINE KEY GOLD W AS "DW."

Attempt to CUT or APPEND to current buffer

This message occurs in keypad mode when you use CUT or APPEND and the current buffer is the PASTE buffer. This message also occurs in nokeypad mode whenever you specify the current buffer name with the CUT or APPEND command. PASTE is the default buffer for the nokeypad CUT and APPEND commands.

Attempt to PASTE current buffer

This message occurs in keypad mode when you use PASTE and the current buffer is the PASTE buffer. This message occurs in nokeypad mode whenever you specify the current buffer with the PASTE command. PASTE is the default buffer for the nokeypad PASTE command.

Attempt to re-enter EDT

In using callable EDT, you have attempted to recall EDT without having exited.

Backup past top of buffer

Whenever you attempt to move the cursor ahead of the first line in your file (the top of the buffer), EDT displays this warning.

Cannot set terminal type from change mode

You have tried to issue a SET TERMINAL {VT100 | VT52 | HCPY} command while working in a screen mode (using either the keypad COMMAND function or the nokeypad EXT command). In order to change the terminal type with the SET TERMINAL command, you must shift to line mode.

Change mode can be entered only from a terminal

You cannot enter change mode (keypad, nokeypad, or hardcopy change mode) from a network, a batch command file, or a startup command file.

Command buffer exhausted

You have attempted to string too many nokeypad mode commands together on one line. EDT can only handle 255 characters per line.

Command buffer exhausted during XLATE command processing

You have put too many characters in your XLATE procedure. EDT can only handle 255 characters per command line.

Command file could not be closed

The startup command file cannot be closed. The appropriate system message is also displayed when this problem occurs so that you can have more information on the nature of the problem.

Command file could not be opened

The startup command file cannot be opened. The appropriate system message is also displayed when this problem occurs so that you can have more information on the nature of the problem.

Command file does not exist

You have asked EDT to use a startup command file that does not exist in the specified directory.

Consistency check failed, please check your file

This message indicates a problem with EDT itself, not with any input you have given EDT. When EDT's consistency check fails, it could mean that there are omissions or other problems in the output file. Always report this message to your system manager.

Could not align tabs with cursor

This message refers to the tab compute functions in the screen modes. EDT issues it whenever you use the keypad CTRL/A function or the nokeypad TC command and the cursor position is not evenly divisible by the SET TAB value.

CTRL/C ignored

You have tried to use CTRL/C to interrupt an EDT operation, but EDT does not allow you to abort that operation.

Destination for MOVE or COPY not found

You have asked EDT to move or copy text to a location that it could not find.

Editor aborted

For some reason, EDT experienced some difficulties and had to abort the session. You can use the journal file that EDT saves to recover your editing work.

Entity must be WORD, SENTENCE, PARAGRAPH, or PAGE

The SET ENTITY and SHOW ENTITY commands require that you include one of these four entity options. EDT cannot accept any other entities.

Error in command option

You have used an invalid qualifier with your command.

Error in range specification

You have used an invalid range specifier. Either the specifier has been entered incorrectly or you have used a combination of specifiers one of which is unacceptable.

Error opening terminal for input

EDT is unable to access the specified terminal for input. This message appears when you are using EDT from a running program.

Error opening terminal for output

EDT is unable to access the specified terminal for output. This message appears when you are using EDT from a running program.

Error reading from input file

EDT was unable to read an input file when it started the editing session. The problem file could be the main input file, the journal file if you are using the /RECOVER specifier, or the startup command file if you are using one.

Error reading from terminal

EDT is unable to process the input it is receiving from the terminal. The message appears when you are using EDT from a running program.

Error writing to output file

EDT was unable to produce an output file. The problem file could be the MAIN buffer copy, the journal file, or a file requested with the PRINT or WRITE command.

File attributes error

The file cannot be edited by EDT because of its attributes. (For example EDT cannot edit an indexed file on a RSTS/E system.)

File name:

This message accompanies another message. It supplies the name of the file referred to by the other message.

File specification required

You have not specified a file name (or file specification) with an INCLUDE, PRINT, or WRITE command.

For help on any other keypad key, press the key

You can ask for help information on another keypad key without having to press the HELP key again.

Help file could not be closed

In using the SET HELP command, you have specified the name of a HELP file that EDT was unable to close. Some problem exists at the system level that prevents EDT from closing the file.

Help file could not be opened

In using the SET HELP command, you have specified the name of a HELP file that cannot be opened because it does not exist, you do not have access to it, or it is not properly installed on your system.

Help File Index could not be initialized

You have created a new EDT HELP file or modified an existing EDT HELP file, but in doing so, you have either neglected to include an index number or have placed an indexed entry out of the proper nested order.

Include file could not be closed

In using the INCLUDE command, you have specified the name of a file that EDT was unable to close. Some problem exists at the system level that prevents EDT from closing the file.

Include file could not be opened

In using the INCLUDE command, you have specified the name of a file that cannot be opened. Some problem exists at the system level that prevents EDT from opening the file.

Include file does not exist

You have typed a file specification with the INCLUDE command for a file that does not exist. The file might exist in a directory other than the one you specified or implied with the command. Or you might have made a typographical error in the file specification.

Include file input record too large, truncated to 255 characters

EDT can only handle 255 characters on a line. The file you want to include has one or more lines in it that exceed 255 characters. EDT truncates the line to be only 255 characters long.

Input file could not be closed

EDT was unable to close the input file that you named in your EDIT/EDT command line or in the line mode INCLUDE command line. Some problem exists at the system level that prevents EDT from closing the file.

Input file could not be opened

EDT was unable to access the input file you named in your EDIT/EDT command line. The problem is caused by either faulty syntax in the command statement or specification of a nonexistent directory.

Input file does not exist

You have asked EDT to edit a file that does not exist in your current directory or the directory you specified. Either you have given an insufficient or incorrect file specification in your EDIT/EDT command line, or you are using EDT to create a new file.

Input file does not have standard text file format

You are editing a file that does not have standard file attributes (for example, a file that does not have carriage returns at the ends of lines). This message is only a warning. EDT allows you to edit the file, but be aware that difficulties can arise if you attempt to edit the non-standard elements.

Insufficient memory

There is not enough computer memory to complete the processing of the last command. This message is rarely, if ever, seen. It could occur if you try to create a new text buffer or use the DEFINE KEY command after using EDT for hours in a single editing session.

Internal software error - please submit an SPR

You are experiencing a problem with the EDT software as installed on your system. Please send in a Software Performance Report. This message is rarely, if ever, seen.

Invalid buffer name

You have used an invalid character in a buffer name or have not started your buffer name with a letter.

Invalid character

You have attempted to insert a character by using the SPECINS function or ASC command, but have not supplied a valid decimal equivalent for the character.

Invalid entity

You have specified an entity that EDT does not recognize.

Invalid option for that command

You have used a qualifier that is not valid for the command.

Invalid parameter for SET or SHOW

You have used a parameter with your SET or SHOW command that is not valid for the command.

Invalid string

You have specified a string that EDT cannot handle. For example, this message occurs when you use a search string longer than 64 characters.

Invalid subcommand

You have typed an invalid nokeypad command.

Invalid value in SET command

You have supplied an invalid specifier with a SET command.

I/O error on work file

EDT is unable to access the text storage area in which it holds buffers. An additional message explains the problem, for example, **Device full** or **Device write locked**.

Journal file could not be closed

You have used the /RECOVER qualifier in your EDIT/EDT command line, but EDT was unable to close the journal file after processing the commands in it. Some problem exists at the system level with that file.

Journal file could not be opened

You have used the /RECOVER qualifier in your EDIT/EDT command line, but EDT was unable to open the journal file. Either you supplied the name of a nonexistent file, or you did not have access to the file that you wanted to use.

Keys cannot be defined in Nokeypad mode

The nokeypad DEFK command is available solely for the purpose of reassigning the key definition function in keypad mode to a key other than CTRL/K. It cannot be used to redefine keys.

Line exceeded 255 characters, truncated

You have typed more than 255 characters on a line without using the RETURN key to send the characters to EDT. EDT can handle up to 255 characters per line.

____ lines copied

This message tells you that either **No lines** have been copied or displays the exact number of lines that have been copied.

____ lines deleted

This message tells you that either **No lines** have been deleted or displays the exact number of lines that have been deleted.

____ lines moved

This message tells you that either **No lines** have been moved or displays the exact number of lines that have been moved.

MACRO or KEY required

After the DEFINE command word you must type either KEY or MACRO.

Max input line of 2814749767 exceeded, file input terminated

You are attempting to put more than 2814749767 lines into a buffer. EDT stops accepting additional lines when it reaches its limit.

Max line number exceeded – lines no longer ascending; resequence recommended

EDT has reached the limits of its line numbering capacity. If you continue to add lines to your buffer, EDT is no longer able to number them uniquely. You should resequence the buffer if you have fewer lines than line numbers.

Max number of lines for this buffer exceeded

You cannot add any more lines to this buffer. However, you can place any remaining lines in a different buffer.

No definition

The key name you supplied with the SHOW KEY command has no definition at this point in your editing session.

No help available for that key

There is no information in the current EDT HELP file on the use of the key you have pressed. If you have another HELP file available on your system that contains the information, use the SET HELP command to gain access to that HELP file.

No more than 65535 lines can be processed in a single command

You have specified a range that exceeds 65535 lines.

No output file written

You have attempted to put text into an output file, but EDT cannot handle the record format of that text. You must change the record format of the file, if possible, before editing it.

No select range active

You have used a command that requires a select range, but did not establish the select range before issuing the command.

No such line

You have used a nonexistent line number as a range specifier.

Now enter the definition terminated by ENTER

When you are defining or redefining keys with the keypad CTRL/K function, this message appears as soon as you press the key or key sequence you want to define.

Numeric value illegal

You have specified a number that is too large for the command specifier, for example, SET SCREEN 999.

Numeric value required

The command you issued must have a numeric value entered at the point indicated by the circumflex (^).

ORIGINAL line numbers no longer an EDT feature

EDT has been revised to enable users to edit sequence-numbered files and to retain the sequence numbers for future work. The new capabilities replace the original line number feature.

Output file could not be closed

EDT has opened the output file and has copied the MAIN buffer text into that file. However, for some reason at the system level, EDT is unable to close that file.

Output file could not be created

EDT was unable to create the output file with the file specification you supplied with the EXIT command. This can be caused by problems in the file name itself (either it has too many characters or an invalid character). Other causes include problems in the directory specification, if you used one (either the directory does not exist, you do not have access to it, or the directory specification format is incorrect). This message could also mean that there is a problem with the file system, such as insufficient disk space.

Parenthesis mismatch

The number of left parentheses [(] in your nokeypad command does not equal the number of right parentheses [)].

Parsing stack overflow

Your command is too complex for EDT to process. If you have given a valid command, break it into several smaller commands.

Please answer Y(es), N(o), Q(uit) or A(ll)

You are using the /QUERY qualifier with a line mode command and have not given one of the four acceptable responses to the question mark (?) prompt.

Pass bad status to caller

A problem has occurred in trying to use callable EDT. The bad status has not been passed to the calling program due to an error in EDT. Submit a Software Performance Report (SPR).

Press return to continue

You have issued a line mode command from either keypad or nokeypad mode. After processing that command, EDT instructs you to press the RETURN key to continue your editing work.

Press the key you wish to define

This message appears when you are using the keypad CTRL/K function to define a key. When you see the message, press the key or key sequence you want to define or redefine.

Print file could not be closed

EDT has opened the output file and copied the text specified with the PRINT command into that file. However, for some reason at the system level, EDT is unable to close that file.

Print file could not be created

EDT was unable to create the output file with the file specification you supplied with the PRINT command. This can be caused by problems in the file name itself (either it has too many characters or an invalid character). Other causes include problems in the directory specification, if you used one (either the directory does not exist, you do not have access to it, or the directory specification format is incorrect). This message could also mean that there is a problem with the file system, such as insufficient disk space.

Quoted string required

The command you have issued requires that you supply a string enclosed in quotation marks. The circumflex (^) indicates where the quoted string should go. Either single (') or double (") quotes can be used.

Range for RESEQUENCE must be contiguous

The range specification for the RESEQUENCE command cannot contain any gaps. All lines must be adjacent.

Range specified by /SEQUENCE would cause duplicate or non-sequential numbers

You have asked EDT to RESEQUENCE some lines such that some line numbers in your buffer would be duplicated or would be out of sequential order.

Record too big, truncated to 255 characters

The file you are attempting to edit or include has lines longer than 255 characters. EDT has deleted all characters from position 256 on. EDT's line length is limited to 255 characters.

Search string cannot be null

You have accidentally asked EDT to search for nothing.

Select complete lines only

You are using the SELECT line range specifier with a line mode command such as COPY, FILL, MOVE, PRINT, TAB ADJUST, or WRITE. However, the select range that is currently active has at least one partial line. The select range must include only complete lines when it is used with line mode commands.

Select range is already active

You are trying to establish a select range when one is already in effect.

Sequence increment must be less than 65536

You have given too large a number as the :increment specifier for the /SEQUENCE qualifier.

Sequence number must be less than 65536

You have given too large a number as the :integer specifier for the /SEQUENCE qualifier.

String delimiter must be non-alphanumeric

EDT thinks you are using a letter, digit, or percent sign (%) as a string delimiter.

String was not found

EDT could not find the string you wanted to locate when searching in the current direction. Either the string does not exist in the current buffer, or it is in a part of the buffer that was not accessed by the search process.

____ substitutions made

This message tells you that either **No** substitutions have been made, or displays the exact number of substitutions that have been successfully processed.

That key is not definable

EDT does not allow you to define the key or key sequence you have specified for DEFINE KEY (line mode) or CTRL/K (keypad mode).

To exit from HELP, press the spacebar

Press the spacebar to resume your editing session after you have finished using the keypad HELP facility.

To return to the keypad diagram, press the return key

EDT displays this message when you are using the keypad HELP facility. It reminds you to press the RETURN key when you want to display the keypad diagram on your screen after seeing a description of a keypad editing function.

Unexpected characters after end of command

EDT is unable to process your command because it did not end properly. Often this means that you omitted something such as a space, quotation marks, or delimiters, or else you typed something new without first pressing the RETURN key.

Unrecognized command

EDT does not recognize the command you have typed. This message appears in a variety of circumstances. Check your command line to determine the problem.

Unrecognized command option

You have used a qualifier that EDT does not recognize.

Work file failed to close

EDT is unable to close its work file.

Work file failed to open

EDT is unable to open its work file.

Work file overflow

You have exceeded EDT's limit on the size of its work file. The limit is 65535 blocks. To avoid this problem, use the CLEAR command to delete a buffer when it is no longer needed. This precaution will also help to reduce the size of the work file on systems with small disks.

Working

EDT is busy processing your command, but needs a few seconds to complete the job. The message assures you that EDT is working on the last command you issued. This message does not appear on RSTS/E systems.

Write file could not be closed

EDT has opened the output file and copied the text specified with the WRITE command into that file. However, for some reason at the system level, EDT is unable to close that file.

Write file could not be created

EDT was unable to create the output file with the file specification you supplied with the WRITE command. This can be caused by problems in the file name itself (either it has too many characters or an invalid character). Other causes include problems in the directory specification, if you used one (either the directory does not exist, you do not have access to it, or the directory specification format is incorrect). This message could also mean that there is a problem with the file system, such as insufficient disk space.

Appendix B
EDT Command Summary

B.1 KEYPAD COMMANDS

Keypad Function	VT100	VT52
ADVANCE	4	4
APPEND	9	blue color + ←
BACKSPACE [CTRL/H] [F12]	BACK SPACE	BACK SPACE
BACKUP	5	5
GOLD BOTTOM	PF1 + 4	blue color + 4
CHAR (character)	3	—
GOLD CHNGCASE (change case)	PF1 + 1	blue color + 1
GOLD COMMAND	PF1 + 7	blue color + 7
CTRL/A (tab compute)	CTRL + A	CTRL + A
CTRL/C (abort EDT operation)	CTRL + C	CTRL + C
CTRL/D (tab decrement)	CTRL + D	CTRL + D
CTRL/E (tab increment)	CTRL + E	CTRL + E
CTRL/H (move to beginning of line)	CTRL + H	CTRL + H

Keypad Function	VT100	VT52
CTRL/I (tab)	CTRL + I	CTRL + I
CTRL/J (delete to beginning of word)	CTRL + J	CTRL + J
CTRL/K (define key)	CTRL + K	CTRL + K
CTRL/L (insert form feed)	CTRL + L	CTRL + L
CTRL/M (insert carriage return)	CTRL + M	CTRL + M
CTRL/R (refresh screen)	CTRL + R	CTRL + R
CTRL/T (tab adjust)	CTRL + T	CTRL + T
CTRL/U (delete to beginning of line)	CTRL + U	CTRL + U
CTRL/W (refresh screen)	CTRL + W	CTRL + W
CTRL/Z (shift to line mode)	CTRL + Z	CTRL + Z
CUT [Remove]	6	3
DEL C (delete character)	,	6
GOLD DEL EOL (delete to end of line)	PF1 + 2	blue color + 2
DELETE [X]	DELETE	DELETE
DEL L (delete line)	PF4	gray color
DEL W (delete word)	-	9
DO (LK201 only)	DO	
DOWN (down arrow)	↓	↓
ENTER	ENTER	ENTER

Keypad Function	VT100	VT52
EOL (end of line)	2	2
GOLD FILL	PF1 + 8	CTRL + F
GOLD FIND [Find]	PF1 + PF3	blue color + 8
FNDNXT (find next)	PF3	8
GOLD	PF1	blue color
HELP [Help]	PF2	red color
LEFT (left arrow)	←	←
LINE	0	0
LINEFEED [CTRL/J] [F13]	LINE FEED	LINE FEED
GOLD OPEN LINE	PF1 + 0	blue color + 0
PAGE	7	7
GOLD PASTE [Insert Here]	PF1 + 6	blue color + 3
GOLD REPLACE	PF1 + 9	blue color + ↑
GOLD RESET	PF1 + .	blue color + .
RETURN	RETURN	RETURN
RIGHT (right arrow)	→	→
SECT (section)	8	blue color + ↓

+SECT (section forward) (LK201 only) [Next Screen]

–SECT (section backward) (LK201 only) [Prev Screen]

Keypad Function	VT100	VT52
SELECT [Select]	.	.
GOLD SPECINS (special insert)	PF1 + 3	blue color + →
GOLD SUBS (substitute)	PF1 + ENTER	blue color + ENTER
TAB [CTRL/I]	TAB	TAB
GOLD TOP	PF1 + 5	blue color + 5
GOLD UND C (undelete character)	PF1 + ,	blue color + 6
GOLD UND L (undelete line)	PF1 + PF4	blue color + gray color
GOLD UND W (undelete word)	PF1 + -	blue color + 9
UP (up arrow)	↑	↑
WORD	1	1

B.2 LINE COMMANDS

```
CHANGE [=buffer] [range] [;nokeypad-command(s)]

CLEAR buffer

COPY [=buffer-1] [range-1] TO [=buffer-2] [range-2] [/QUERY] [/DUPLICATE:n]

CTRL/C   (abort EDT operation)

CTRL/R   (refresh line)

CTRL/Z   (conclude inserting)

DEFINE KEY key-name AS "string"

DEFINE MACRO macro-name

DELETE [=buffer] [range] [/QUERY]

EXIT [file-specification] [/SEQUENCE[:initial[:increment]]] [/SAVE]

FILL [=buffer] [range]

FIND [=buffer] [range]

HELP [topic [subtopic ... ] ]

INCLUDE file-specification [=buffer] [range]

INSERT [=buffer] [range] (RETURN) text (CTRL/Z)
INSERT [=buffer] [range] ;line-to-be-inserted

MOVE [=buffer-1] [range-1] TO [=buffer-2] [range-2] [/QUERY]

<null>   (implied TYPE)
   [=buffer] [[%]range]

PRINT file-specification [=buffer] [range]

QUIT [/SAVE]

REPLACE [=buffer] [range] (RETURN) text (CTRL/Z)
REPLACE [=buffer] [range] ;line-to-be-inserted

RESEQUENCE [=buffer] [range] [/SEQUENCE[:initial[:increment]]]

SET AUTOREPEAT
SET NOAUTOREPEAT

SET CASE NONE
SET CASE LOWER
SET CASE UPPER

SET COMMAND file-specification

SET CURSOR top:bottom
```

```
SET ENTITY WORD "string"
SET ENTITY SENTENCE "string"
SET ENTITY PARAGRAPH "string"
SET ENTITY PAGE "string"

SET FNF
SET NOFNF

SET HELP [file-specification]

SET KEYPAD
SET NOKEYPAD

SET LINES number

SET MODE CHANGE
SET MODE LINE

SET NUMBERS
SET NONUMBERS

SET PARAGRAPH WPS
SET PARAGRAPH NOWPS

SET PROMPT prompt-type "string"

SET QUIET
SET NOQUIET

SET REPEAT
SET NOREPEAT

SET SCREEN width

SET SEARCH GENERAL
           EXACT
           WPS
           CASE INSENSITIVE
           DIACRITICAL INSENSITIVE
SET SEARCH BEGIN
           END
SET SEARCH BOUNDED
           UNBOUNDED

SET SUMMARY
SET NOSUMMARY

SET TAB number
SET NOTAB

SET TERMINAL HCPY
             VT100
             VT52
SET TERMINAL SCROLL
             NOSCROLL
SET TERMINAL EIGHTBIT
             NOEIGHTBIT
SET TERMINAL EDIT
             NOEDIT
```

```
SET TEXT END "string"
SET TEXT PAGE "string"

SET TRUNCATE
SET NOTRUNCATE

SET VERIFY
SET NOVERIFY

SET WORD DELIMITER
SET WORD NODELIMITER

SET WRAP n
SET NOWRAP

SHOW AUTOREPEAT

SHOW BUFFER

SHOW CASE

SHOW COMMAND

SHOW CURSOR

SHOW ENTITY WORD
SHOW ENTITY SENTENCE
SHOW ENTITY PARAGRAPH
SHOW ENTITY PAGE

SHOW FILES

SHOW FNF

SHOW HELP

SHOW KEY key-name

SHOW KEYPAD

SHOW LINES

SHOW MODE

SHOW NUMBERS

SHOW PARAGRAPH

SHOW PROMPT prompt-type

SHOW QUIET

SHOW REPEAT

SHOW SCREEN

SHOW SEARCH

SHOW SUMMARY
```

```
SHOW TAB

SHOW TERMINAL

SHOW TEXT END
SHOW TEXT PAGE

SHOW TRUNCATE

SHOW VERIFY

SHOW VERSION

SHOW WORD

SHOW WRAP

SUBSTITUTE/[string-1]/string-2/ [=buffer] [range]
       [/BRIEF[:n]] [/QUERY] [/NOTYPE]

[SUBSTITUTE] NEXT[/[string-1]/string-2/]

TAB ADJUST [-]n [=buffer] [range]

TYPE [=buffer] [range] [/BRIEF[:n]] [/STAY]

WRITE file-specification [=buffer] [range] [/SEQUENCE[:initial[:increment]]]
```

B.3 NOKEYPAD COMMANDS

ADV (advance)

APPEND
```
[+|-][count]APPEND[+|-][count]entity[=buffer]
```

ASC (ASCII)
```
[number]ASC
```

BACK (backup)

BELL

CHGC (change case)
```
[+|-][count]CHGC[+|-][count]entity
```

CHGL (change case lower)
```
[+|-][count]CHGL[+|-][count]entity
```

CHGU (change case upper)
```
[+|-][count]CHGU[+|-][count]entity
```

^ (circumflex – used to enter ASCII control characters)
```
[count]^character
```

CLSS (clear search string)

CTRL/C (abort EDT operation)

CTRL/Z (conclude inserting)

CUT
```
[+|-][count]CUT[+|-][count]entity[=buffer]
```

D (delete)
```
[+|-][count]D[+|-][count]entity
```

DATE (insert date and time)

DEFK (define key)

DESEL (deactivate select)

DLWC (default lowercase)

DMOV (default move)

Down Arrow

DUPC (default uppercase)

EX (exit to line mode)

EXT (extend)
```
EXT line-mode-command(s)
```

FILL
```
[+|-][count]FILL[+|-][count]entity
```

```
HELP

I   (insert)
    Itext^Z
    I (RETURN) text (CTRL/Z)

KS   (KED substitute)

Left Arrow

"move"
    [+|-][count]entity

PASTE
    [count]PASTE[=buffer]

QUIT

R   (replace)
    [+|-][count]R[+|-][count]entity

REF   (refresh)

Right Arrow

S   (substitute)
    [+|-][count]S/[string-1]/string-2/

SEL   (select)

SHL   (shift left)
    [count]SHL

SHR   (shift right)
    [count]SHR

SN   (substitute next)
    [+|-][count]SN

SSEL  (search and select)
    [+|-]SSEL[+|-]"string"

TAB
    [count]TAB

TADJ   (tab adjust)
    [+|-][level-count]TADJ[+|-][entity-count]entity

TC   (tab compute)

TD   (tab decrement)
    [count]TD

TGSEL  (toggle select)
```

TI (tab increment)
[count]TI

TOP

UNDC (undelete character)
[count]UNDC

UNDL (undelete line)
[count]UNDL

UNDW (undelete word)
[count]UNDW

Up Arrow

XLATE (translate)
XLATEstring^Z

Appendix C
Terminal Support and Interface for EDT

C.1 Terminals That EDT Supports

EDT supports all Digital asynchronous terminals for line editing.

EDT supports the following screen terminals for screen editing (both keypad and nokeypad):

VT100
VT52

The following terminals, when they are operating as VT100s, are also supported for screen modes:

VT101	VT125	VT131
VT102		VT132

Terminals using the LK201 keyboard, in both VT100 and VT200 modes

The GIGI (VK100) terminal is supported as a VT100 subset terminal with no scrolling regions and a screen width of 84 characters.

There are two restrictions for VT101s and other VT100-type terminals that do not have the advanced video option (AVO):

1. The cursor must be set in block mode.
2. The only screen width setting allowed is 80.

EDT's SET AUTOREPEAT feature enables EDT to use the DECARM VT100 control sequence to prevent keypad keys (including the arrow keys) from repeating faster than EDT can update the screen. However, on some VT100-type terminals, SET AUTOREPEAT causes the arrow keys to repeat very slowly (approximately .5 seconds for each keystroke). For these terminals, you might want to use the SET NOAUTOREPEAT command to disable the DECARM VT100 control sequence. With SET NOAUTOREPEAT, EDT simply skips intermediate screen updates if it gets behind. (SET NOAUTOREPEAT is a line mode command.)

If you have an add-on printer port with your VT100-type terminal, EDT's NOAUTOREPEAT setting is not compatible with the firmware in that terminal. You must have your EDT session use the default AUTOREPEAT setting.

C.2 Terminal Interface

When you call up EDT to edit a file, EDT checks with your operating system to find out what type of terminal you are using. In most cases, EDT is able to learn everything it needs to know about your terminal, so you do not need to supply any additional information.

If you are using the VAX/VMS operating system, EDT gets all of the following information from the system:

terminal type
presence of scrolling regions
seven- or eight-bit capacity
screen editing capability
screen width (line length)

Not all of this information is available from RSTS/E and RSX-11M/M-PLUS systems. EDT uses defaults for any missing information. You can use EDT's SHOW TERMINAL command to find out which terminal settings EDT is using. The SHOW TERMINAL command (a line mode command) displays one setting from each group:

{VT100 | VT52 | hardcopy}
{scroll | noscroll}
{eightbit | noeightbit}
{edit | noedit}

The SET TERMINAL EDIT setting is for terminals such as the VT102 and VT131 that have internal editing features built into the terminals themselves. EDT makes use of the IL (insert line), DL (delete line), the insert state of IRM (insertion-replacement mode), and DCH (delete character) features. EDT does not use the internal block transmission mode of terminals such as the VT131.

If any terminal settings are not accurate, you can use the SET TERMINAL command to make the necessary adjustments. If you are making an adjustment in the first group of parameters – terminal type (VT100, VT52, or HCPY), you must be in line mode when you issue the SET TERMINAL command.

The SHOW SCREEN command tells you what screen width (or line length) EDT is using. To change that setting, use the SET SCREEN command. For VT100-type terminals without advanced video option (AVO) and for all VT52 terminals, the only valid screen width for screen editing is 80. For the VK100, the only valid screen width is 84. For all other VT100-type terminals the two valid screen widths for screen editing are 80 and 132.

If your system is consistently giving EDT the wrong information about your terminal, you can put the appropriate SET TERMINAL or SET SCREEN command in your EDT startup command file. Doing this saves you from having to type the command each time you call up EDT to edit a file. However, if you use your startup command file with a variety of terminals, you will have to issue the SET command(s) whenever you work at a different terminal.

C.3 Terminals Operating at Low Data Transmission Rates

Keypad editing is designed to work at high data transmission rates such as 9600 bits per second (960 characters per second). When you use EDT at a low data transmission rate (for example 300 bits per second), EDT is unable to display characters as rapidly as you can type them.

If your terminal is operating at a low data rate, you might find it more convenient to use line or nokeypad mode. These modes are more efficient at slower speeds than keypad editing.

These are things you can do to improve your keypad editing sessions at a slow terminal:

- Whenever possible, insert text on new lines or at the ends of lines.
- Reduce the number of lines that appear on the screen with the SET LINES command and reset the scrolling points for the cursor with the SET CURSOR command so that the cursor operates only in the reduced screen area.

SET LINES and SET CURSOR commands such as these speed up keypad editing at low data rates:

```
SET LINES 10
SET CURSOR 2:7
```

Appendix D
Using EDT with the VAX/VMS Operating System

This appendix contains information about using EDT with the VAX/VMS operating system. You can find out about the procedures for starting an EDT session, EDT's use of files, EDIT /EDT command qualifiers, control characters that should not be defined, and special terminal settings. The final section contains information about calling EDT from a running program.

D.1 Calling Up EDT

Since EDT is the default editor for the VAX/VMS system, you do not need to specify the /EDT qualifier with the EDIT command. The syntax for the VAX/VMS system command that calls up EDT is:

```
$ EDIT [/QUALIFIER(s)]
_File:  file-specification

$ EDIT [/QUALIFIER(s)] file-specification
```

The prompt _File: appears whenever you do not supply a file name on the EDIT command line. The file specification must include at least the file name. No wildcard characters can be used. File specifications can include the full file specification elements as needed, for example:

```
$ EDIT DISK$USER:[SMITH]FILE.DAT
```

D.2 Information on Using Files with EDT

You do not need to include the version number as part of the file specification. When you omit the version number, EDT accesses the highest version number in your directory. If you want to edit an earlier version, you must include the number of that version in the file specification.

Unless you specify the name of an output file either with the /OUTPUT qualifier to your EDIT /EDT command line or with the line mode EXIT command, EDT uses the same file name for the output file as the file you are editing. The original version of the file you are editing is not

altered in any way. It remains in its directory. When you give the line mode EXIT command, EDT creates a new file in that directory using the same name as the input file, but with the highest version number of any file with the same file specification.

```
$ EDIT CHAPTER1.RNO;7
   .
   .
   .
*EXIT
DISK$USER:[SMITH]CHAPTER1.RNO;8     275 lines

$ EDIT [JONES]CHAPTER2.RNO;1
   .
   .
   .
*EXIT
DISK$USER:[JONES]CHAPTER2.RNO;2     3790 lines
```

If you are editing a file from another directory and want to put the edited version in your current directory, simply type the file name and file type with EXIT command.

```
$ EDIT [JONES]CHAPTER2.RNO;1
   .
   .
   .
*EXIT CHAPTER2.RNO
DISK$USER:[SMITH]CHAPTER2.RNO;1     3790 lines
```

Notice that the version number is one. This is because the new file is the only version of CHAPTER2.RNO that exists in the SMITH directory. The original is located in the JONES directory.

D.3 EDIT /EDT Command Qualifiers

The VAX/VMS operating system uses several qualifiers with the EDIT /EDT command line. Table D-1 lists all the qualifiers and their defaults. Explanations of the qualifiers are given after the table. The examples show the qualifiers typed directly after the EDIT command, before the input file specification. You can place the qualifiers anywhere on the command line after the initial EDIT /EDT command.

File specifications are separated from the qualifier name by either an equal sign (=) or a colon (:), for example:

```
/COMMAND=EDTINIX.EDT

/COMMAND:EDTINIX.EDT
```

Table D.1 VAX/VMS EDT Command Qualifiers

Command Qualifiers	Default
/[NO]COMMAND [=command-file]	/COMMAND=SYS$LIBRARY:EDTSYS.EDT
/[NO]CREATE	/CREATE
/[NO]JOURNAL [=journal-file]	/JOURNAL=file name.JOU
/[NO]OUTPUT [=output-file]	/OUTPUT=file name.type;n+1
/[NO]READ_ONLY	/NOREAD_ONLY
/[NO]RECOVER	/NORECOVER

/COMMAND=command-file
/NOCOMMAND

Determines whether EDT uses a startup command file or not. The /COMMAND qualifier is followed by an equal sign (=) and then the specification of the startup command file. **.EDT** is the default file type for startup command files.

```
$ EDIT /COMMAND=XEDTINI.EDT MEMO.DAT
```

If you do not include the /COMMAND=command-file qualifier, EDT looks for logical name EDTSYS to read the system-wide command file. If this logical name is not defined, EDT looks next for the logical EDTINI defined in your LOGIN.COM file and then for the file EDTINI.EDT in your default directory. If none of these exists and you have not used the /COMMAND qualifier, EDT begins your editing session in the default state.

When you do not want EDT to process either the system-wide startup command file or the EDTINI.EDT file in your default directory, use the /NOCOMMAND qualifier.

```
$ EDIT/NOCOMMAND MEMO.DAT
```

See Chapter 7 for information on how to create and use startup command files.

/CREATE
/NOCREATE

Controls whether EDT creates a new file when the specified input file is not found. Normally, EDT creates a new file to match the input file specification if it cannot find that file name in the specified directory. When you use /NOCREATE in the EDT command line and accidentally type a specification for a file that does not exist, EDT prints an error message and returns you to DCL command level.

```
$ EDIT /EDT /NOCREATE LETTERX.RNO
Input file does not exist
$
```

/JOURNAL=journal-file
/NOJOURNAL

Determines whether EDT keeps a journal file during your editing session. The default file name for the journal file is the same as the input file name. EDT uses **.JOU**as the default file type. The /JOURNAL qualifier enables you to use a different file specification for the journal file, for example:

```
$ EDIT /JOURNAL=SAVE MEMO.DAT
```

If you are editing a file from another directory and want the journal file to be located in that directory, you must use the /JOURNAL qualifier with a file specification that includes the directory name. Otherwise, EDT creates the journal file in the current directory.

If you include a directory name in the journal file specification, that directory should not be write protected.

If you do not want EDT to keep a record of your editing session operations, use the /NOJOURNAL qualifier in the EDT command line.

```
$ EDIT /NOJOURNAL MEMO.DAT
```

You must use the /RECOVER qualifier to have EDT process the commands contained in the journal file. Details on using EDT's journal facility appear in Chapters 2 and 7.

/OUTPUT=output-file
/NOOUTPUT

Determines whether EDT creates an output file at the end of your editing session. An equal sign (=) separates the /OUTPUT qualifier from the output file specification. By default, the file specification for the output file is the same as that of the input file. Use the /OUTPUT qualifier to have the MAIN buffer text put in a file with a different specification.

```
$ EDIT /OUTPUT=OUTMEM.DAT MEMO.DAT
```

You can include directory information as part of your output file specification to send output to another directory.

```
$ EDIT /OUTPUT=[SMITH.MAIL]MEMO.DAT MEMO.DAT

$ EDIT /OUTPUT=[JONES]FROMBROWN.DAT MEMO.DAT
```

The /NOOUTPUT qualifier suppresses the creation of an output file, but not the creation of a journal file. If you are testing some edits and are not sure you want an output file, you can use the /NOOUTPUT qualifier in the EDIT command line. A system interruption will not prevent you from recreating your editing session because a journal file is still being maintained. If you later decide that you want to keep your editing work, you

can save the edited text in spite of the /NOOUTPUT qualifier by using the line mode WRITE command to put the text in an external file before you end the session.

```
$ EDIT /NOOUTPUT MEMO.DAT
   .
   .
   .
*WRITE MEMO1.DAT
DISK$USER:[SMITH]MEMO1.DAT;1 27 lines
*QUIT
$
```

/READ_ONLY
/NOREAD_ONLY

Determines whether EDT keeps a journal file and creates an output file. With /NOREAD_ONLY, the default, EDT maintains the journal file in case a system interruption occurs and creates an output file when it processes the line mode EXIT command. The /READ_ONLY qualifier has the effect of using both the /NOJOURNAL and /NOOUTPUT qualifiers.

```
$ EDIT /READ_ONLY CALENDAR.DAT
```

Use the /READ_ONLY qualifier when you are merely looking for things in a file. If you are reading files in a library directory, you can use the /READ_ONLY qualifier to avoid the chance of a journal file being left in that directory. If you have set your default directory to one in which you cannot create files, you can use the /READ_ONLY qualifier to inspect files that already exist in that directory.

/RECOVER
/NORECOVER

Determines whether EDT uses the journal file to restore editing work. Normally, EDT starts a session without regard to any journal file that might be in the current directory. This is the default /NORECOVER state. When you use the /RECOVER qualifier, EDT reads the appropriate journal file and processes whatever commands it contains, except a QUIT or EXIT commands. (EDT ignores QUIT and EXIT commands in journal files.) If the journal file type is not **.JOU** or the file name is not the same as the input file name, you must include both the /JOURNAL=journal-file qualifier *and* the /RECOVER qualifier in the EDIT command line.

```
$ EDIT /RECOVER MEMO.DAT

$ EDIT /RECOVER /JOURNAL=SAVE.XXX MEMO.DAT
```

Because /NORECOVER is the default for EDT, you need not specify it in a command line. For more information on the EDT journal/recovery facility, see Chapters 2 and 7.

D.4 Control Characters That Cannot be Defined

When you are defining keys in EDT on a VAX/VMS system, there are several control key sequences that cannot be used by EDT. These control keys already have system functions assigned to them. EDT allows you to define these keys, but when you try to use them, the operating system performs its control function and never passes the control key code to EDT.

For example, suppose you assign the definition **(–W).** to CTRL/S. Later you press CTRL/S to move the cursor back one word. Instead, you find that the terminal output has been suspended, the VAX/VMS function for CTRL/S. (CTRL/Q restores your terminal to normal operation after CTRL/S has been pressed.) When you resume normal operation, you see that pressing CTRL/S had no effect on the cursor position. Other control key sequences might have more serious consequences for your editing session.

These are the control key sequences that you should not try to define:

CTRL/C
CTRL/O
CTRL/P (from the console terminal)
CTRL/Q
CTRL/S
CTRL/T *
CTRL/X
CTRL/Y
CTRL/[

*If you use SET CONTROL = T to enable the DCL control T intercept, do not define CTRL/T in EDT.

D.5 Terminal Information and Settings under the VAX/VMS Operating System

When you call up EDT to edit a file, EDT collects certain information from the VAX/VMS operating system about the type of terminal you are using. This information includes data about the screen width, the number of bits that are used to transmit and receive a character, the existence of adjustable scrolling regions, and the presence of internal editing features in the terminal.

Depending on your type of terminal, EDT makes the following assumptions:

Terminal	Escape Sequences	Scrolling Regions	Autorepeat Suppression
VT100 or VT125	VT100	Yes	Yes
VT100-type DEC_CRT is set	VT100	Yes	No
VT100 subset ANSI_CRT is set	VT100	No	No
VT52	VT52	No	No
"Hardcopy"	None	No	No

EDT takes its information on screen width, eight-bit character representation, and terminal editing features directly from the VAX/VMS system terminal characteristics:

WIDTH EIGHTBIT EDIT

You can use the line mode SET TERMINAL command to correct any misconceptions that your system might be passing to EDT about terminal type, scrolling regions, eightbit capacity, and internal editing features. If your terminal has any of these features, you can use an appropriate SET TERMINAL command to tell EDT not to use the features. However, if your terminal does not have a feature, you cannot use either the system SET TERMINAL command or the EDT SET TERMINAL command to add the feature. If you try to do this, EDT becomes confused and problems will occur in your editing session.

Use the line mode SET SCREEN command to change the number of characters displayed on a line. Hardcopy terminals that take paper 15 inches wide can use a SET SCREEN width of 1 to 132 characters. VT100-type terminals with AVO (advanced video option) can accommodate a width of either 80 or 132 characters per line. The only screen width that you can use with VT52 terminals and VT100-type terminals without AVO is 80. If you are using a VT52 and find that your system is set for a screen width other than 80, use either of the following commands to correct the situation:

```
DCL     $ SET TERMINAL /WIDTH=80

EDT     *SET SCREEN 80
```

When you finish your editing session, EDT always sets VT100-type terminals back to either the 80 or 132 screen width depending on the width that your operating system has set for your terminal.

See Appendix C for more information on terminal types and characteristics.

D.6 Accessing the Same EDTINI File in All Subdirectories

You can have EDT automatically process the same EDTINI.EDT startup command file that exists in your main directory in all your subdirectories. Simply include a definition statement in your LOGIN.COM file to tell EDT where to look for the EDTINI.EDT file.

```
DEFINE EDTINI [main-directory-specification]EDTINI.EDT
```

Now, when you are working in a subdirectory, EDT will find and use the EDTINI.EDT file that is located in your main directory.

D.7 EDT Default File Attributes

EDT carries certain file attributes from the primary input file to the primary output file. If there is no primary input file, EDT uses certain defaults for these file attributes. The WRITE and PRINT commands always create files with the default attributes. The attributes of a file read by the INCLUDE command have no effect on the attributes of the primary output file.

The file attributes that EDT carries have these default values:

Attribute	Default
Protection	same as the process default protection
Organization	sequential
Record Format	variable length, maximum length 255 bytes
Record Attribute	carriage return
If you use the EDT /SEQUENCE qualifier with the EXIT or WRITE command, the Record Format changes:	
Record Format	VFC (Variable with Fixed length Control) with a 2-byte header and a maximum length of 255 bytes

Note that the output file is always noncontiguous.

D.8 Information on EDT HELP Files

Different operating systems use different methods to access data in the EDT HELP file. Since the VAX/VMS system stores the HELP text in a library, organization of the HELP file text has little effect on the accessibility of the information.

In order to extract the HELP file text from the system library, you must use the VAX/VMS LIBRARY command. After you modify the text, use the LIBRARY command again to insert the new version. See your system documentation for more information on the LIBRARY utility.

For more information on how to modify the EDT HELP file text, see Chapter 7 of this manual.

D.9 Work File Location

The work file used by EDT during editing sessions is located on SYS$SCRATCH. If your buffers contain only a small amount of text, EDT holds your text in virtual memory and does not use a work file.

D.10 The Temporary File with EXIT, PRINT, and WRITE

EDT creates a temporary file whenever a system interruption occurs in the middle of an EXIT, PRINT, or WRITE command. The temporary file has the same file name as the file you are editing, but uses **.TMP** as the file type. The **.TMP** file type signals you that the temporary file probably does not contain all the text that you wanted to have in the external file.

In most instances, you will want to delete the **.TMP** file and rerun your EDT session, using the /RECOVER qualifier to process the journal file. The purpose of the **.TMP** file is to make sure that EDT does not add a new file to a directory that contains only part of the text you wanted it to have.

D.11 Callable EDT

EDT can be called on VAX/VMS systems by programs. Calling programs can be written in any VAX–11 language that generates calls using the VAX–11 calling standard.

This section on callable EDT assumes that you know how to call an external facility from the language you are using. This appendix discusses only the parameters that must be passed and returned by EDT. If you need more information on the exact syntax to use with your language, consult the documentation for that language as well as the VAX/VMS system documentation on library procedures.

Callable EDT is a shareable image, which means that you save physical memory and disk space by having all processes access a single copy. The sharable image defines all of the EDT$K and EDT$M symbols, described in this section, as external constants.

You must include a statement in your program accessing the EDT entry point. This reference statement is similar to one for a library procedure. The reference for the EDT entry point is:

```
EDT$EDIT
```

The calling sequence for callable EDT is:

```
status.wlc.v = EDT$EDIT(in_file.rt.dx [,[out_file.rt.dx]
               [,[com_file.rt.dx] [,[jou_file.rt.dx,[options.rv.r]
               [,[fileio.fbpv.rp] [,[workio.fbpv.rp]
               [,[xlate.fbpv.rp] ]]]]]]])
```

Explanations of the parameters follow. The parameter characteristics, listed after the parameter types, indicate restrictions on files and procedures.

status

This is the return status code that indicates the final state of the edit. The three possibilities are: a successful termination, a termination caused by an internal EDT error, or a termination due to problems in one of the procedures (FILEIO, WORKIO, or XLATE). These return status codes are stored in register R0 (zero).

If the EDT session is terminated by EXIT or QUIT, the status will be a success value (bit 0 = 1). If the session is terminated because the file was not found and the /NOCREATE EDIT command qualifier was in effect, the failure code EDT$_INPFILNEX is returned. In the case of an unsuccessful termination caused by an EDT error, a failure code corresponding to that error is returned. Error statuses from FILEIO, WORKIO, and XLATE are explained separately.

in_file

This is the file specification of the input file, passed by descriptor, for EDT to edit. The string that you enter in this calling sequence is passed to the FILEIO routine to open the primary input file. This is the only required parameter.

out_file

This is the file specification of the output file, passed by descriptor. The out_file string is also passed to the FILEIO routine to open the output file for the EXIT command. The default is that the in-file is passed to FILEIO.

com_file

This is the file specification of the command file, passed by descriptor. The com_file string is passed to the FILEIO routine to open the command file. The default is the same as that for EDT command file defaults.

jou_file

This is the file specification of the journal file, passed by descriptor. The jou_file string is passed to the FILEIO routine to open the journal file. The default uses the same file name as in_file but with the **.JOU** file type.

options

This is a bit string of length 32. Only bits 0 through 5 are currently defined; all others must be zero. The default options have all bits set to zero. This is the same as the default setting when you call up EDT to edit a file interactively.

EDT$M_RECOVER (bit 0), if set, causes EDT to read the journal file and execute the commands in it, except for the EXIT or QUIT commands, which are ignored. Once the journal file commands have been processed, editing operates normally. If this bit is set, FILEIO will be asked to open the journal file for both input and output; otherwise FILEIO will only be asked to open the journal file for output. This bit corresponds to the /RECOVER qualifier on the EDT command line.

EDT$M_COMMAND (bit 1), if set, causes EDT$FILEIO to signal if the startup command file cannot be opened. When bit 1 is zero, EDT will intercept the signal from FILEIO indicating that the startup command file could not be opened, and simply proceed with the editing session, without reading any startup command file. If no command file name is supplied with the call to EDT$EDIT, EDT tries to open SYS$LIBRARY:EDTSYS.EDT or, if that fails, EDTINI.EDT. This bit corresponds to the /COMMAND qualifier on the EDT command line. If EDT$M_NOCOMMAND (bit 4) is set, bit 1 has no meaning since bit 4 prevents EDT from trying to open a command file.

EDT$M_NOJOURNAL (bit 2), if set, causes EDT not to open the journal file. This bit corresponds to the /NOJOURNAL or /READ_ONLY qualifier on the EDT command line.

EDT$M_NOOUTPUT (bit 3), if set, causes EDT not to use the input file name as the default output file name. This bit corresponds to the /NOOUTPUT or /READ_ONLY qualifier on the EDT command line.

EDT$M_NOCOMMAND (bit 4), if set, causes EDT not to open a startup command file. This bit corresponds to the /NOCOMMAND qualifier on the EDT command line.

EDT$M_NOCREATE (bit 5), if set, causes EDT to return to the caller if the input file is not found. The status returned is the error code EDT$_INPFILNEX.

fileio

This is the user-supplied routine that is called by EDT to perform file I/O functions. If the user does not need to intercept any file I/O, the entry point EDT$FILEIO can be used for this parameter, or it can be omitted. If you only need to intercept *some* file I/O, call EDT$FILEIO for the other cases.

In order to accommodate routines written in high-level languages that do up-level addressing, this parameter must have a data type of BPV (bound procedure value). A BPV is a two-longword entity in which the first longword contains the address of a procedure entry mask and the second longword is the environment value. When the bound

procedure is called, EDT loads the second longword into R1. If you use EDT$FILEIO for this parameter, set the second longword to zero. You can simply pass a zero for this parameter and EDT will set up EDT$FILEIO as the default and set the environment word to zero.

workio

This is a user-supplied routine that is called by EDT to perform I/O between the work file and EDT. Work file records are addressed only by number and are always 512 bytes long. If the user does not need to intercept work file I/O, you can use the entry point EDT$WORKIO for this parameter or it can be omitted.

In order to accommodate routines written in high-level languages that do up-level addressing, this parameter must have a data type of BPV (bound procedure value). This means that EDT will load R1 with the second longword addressed before calling it. If EDT$WORKIO is used for this parameter, set the second longword to zero. You can simply pass a zero for this parameter and EDT will set up EDT$WORKIO as the default and set the environment word to zero.

xlate

This is a user-supplied routine that EDT calls when it encounters the nokeypad command XLATE. XLATE allows the user of EDT to gain control of the EDT session. If the user does not need to have control of EDT during the editing session, you can use the entry point EDT$XLATE for this parameter, or you can omit it.

In order to accommodate routines written in high-level languages that do up-level addressing, this parameter must have a data type of BPV (bound procedure value). This means that EDT will load R1 with the second longword addressed before calling it. If EDT$XLATE is used for this parameter, set the second longword to zero. You can simply pass a zero for this parameter and EDT will set up EDT$XLATE as the the default and set the environment word to zero.

D.11.1 Calling Sequences for FILEIO, WORKIO, XLATE

The three user-supplied routines FILEIO, WORKIO, and XLATE are each called by EDT using a status return sequence with passed parameters.

FILEIO

```
status.wlc.v = fileio(code.rl.r,stream.rl.r,record.mt.dx,
                      rhb.mt.dx)
```

As in the main calling sequence, the parameters must have the correct characteristics for access type, data type, passing mechanism, and parameter form.

The FILEIO parameters are:

status

This is a VAX/VMS status code that the user's FILEIO routine returns to EDT. The only failure code that is normally returned is RMS$_EOF from a get call. All other RMS errors are signalled, not returned. The RMS signal should include the file name and both longwords of the RMS status. Any errors detected with the file I/O routine can be indicated by setting **status** to an error code. That error code will be returned to the caller of EDT$EDIT. There is a special status value EDT$_NONSTDFIL for nonstandard file opening.

code

This is a code from EDT that specifies what function the user's FILEIO routine is to perform. The valid function codes are:

`EDT$K_OPEN_INPUT (1)`
: The **record** descriptor names a file that is to be opened for input. The **rhb** parameter is the default file name.

`EDT$K_OPEN_OUTPUT_SEQ (2)`
: The **record** descriptor names a file that is to be opened for output as a sequenced file. The **rhb** parameter is the default file name. (For more information on sequenced files, see the section in Chapter 7 on EDT line numbers.)

`EDT$K_OPEN_OUTPUT_NOSEQ (3)`
: The **record** descriptor names a file that is to be opened for output without sequence numbers. The **rhb** parameter is the default file name.

`EDT$K_OPEN_IN_OUT (4)`
: The **record** descriptor names a file that is to be opened for both input and output. The **rhb** parameter is the default file name.

`EDT$K_GET (5)`
: The **record** descriptor is to be filled with data from the next record of the file. If the file has record prefixes, **rhb** is filled with the record prefix. If the file has no record prefixes, **rhb** is not written. When there are no more records in the file, **status** is set to RMS$_EOF.

`EDT$K_PUT (6)`
: The data in the **record** descriptor is to be written to the file as its next record. If the file has record prefixes, the record prefix is taken from the **rhb** descriptor. For a file opened for both input and output, EDT$K_PUT is only valid at the end of the file, indicating that the record is to be appended to the file.

`EDT$K_CLOSE_DEL (7)`
: The file is to be closed and then deleted. The **record** and **rhb** parameters are not used in the call.

`EDT$K_CLOSE (8)`
: The file is to be closed. The **record** and **rhb** parameters are not used in the call.

stream

This is a code from EDT that indicates which file is being used. The valid codes are:

`EDT$K_COMMAND_FILE (1)`
: The command file.

`EDT$K_INPUT_FILE (2)`
: The primary input file.

`EDT$K_INCLUDE_FILE (3)`
: The secondary input file. Such a file is opened in response to an INCLUDE command. It is closed when the INCLUDE command is complete and will be re-used for subsequent INCLUDE commands.

EDT$K_JOURNAL_FILE (4)
: The journal file. If bit 0 of the options is set, it is opened for both input and output and is read completely. Otherwise, it is opened for output only. Once it has been read or opened for output only, it is used for writing. On a successful termination of the editing session, the journal file is closed and deleted. EXIT/SAVE and QUIT/SAVE close the journal file without deleting it.

EDT$K_OUTPUT_FILE (5)
: The primary output file. It is not opened until the the EXIT command is issued.

EDT$K_WRITE_FILE (6)
: The secondary output file. Such a file is opened in response to a WRITE or PRINT command. It is closed when the command is complete and will be re-used for subsequent WRITE and PRINT commands.

record

This parameter is a line of text passed by descriptor from EDT to the user-supplied file I/O routine. The code parameter determines how the record parameter is used. When the code parameter is EDT$K_OPEN, the record parameter is a file name. When the code parameter is EDT$K_GET, the record is a place to store the record that was read from the file. For code parameter EDT$K_PUT, the record is a place to find the record to be written to the file. This parameter is not used if the code parameter is EDT$K_CLOSE.

Note that for EDT$K_GET, EDT uses a dynamic or varying string descriptor. Otherwise, EDT has no way of knowing the length of the record being read. EDT only uses string descriptors that can be handled by the RTL (Run Time Library) routine STR$COPY_DX.

rhb

This parameter also depends on the code parameter. EDT$K_OPEN implies the default file name. When the code is EDT$K_GET and the file has record prefixes, the prefixes are put in the record header buffer. When the code is EDT$K_PUT and the file has record prefixes, the prefixes are taken from the record header buffer. Like the record parameter, EDT uses a dynamic or varying string descriptor for EDT$K_GET and will only use string descriptors that can be handled by the RTL routine STR$COPY_DX.

In summary, when you use EDT$FILEIO for the fileio parameter, files are opened as follows:

- The **record** parameter is always the RMS file name.
- The **rhb** parameter is always the RMS default file name.
- There is no related name for the input file.
- The related name for the output file is the input file with OFP (output file parse). EDT passes either the input file name, the output file name, or the name from the EXIT command, as appropriate, in the record parameter.
- For the journal file name, the input file is the related file with the OFP (output file parse) RMS bit set.
- There is no related file name for the WRITE file. INCLUDE implies input file with OFP.

WORKIO

```
status.wlc.v = workio(code.rl.r,recordno.rl.r,record.bu.dx)
```

As in the main calling sequence, the parameters must have the correct characteristics for access type, data type, passing mechanism, and parameter form.

status

This parameter is a VAX/VMS status code. It is normally a success code, since all RMS errors are generally signalled. The signal parameters should include the file name and both longwords of the RMS status. Any errors detected within work I/O can be indicated by setting **status** to an error code, which will be returned by EDT$EDIT.

code

This parameter refers to the operation to be performed. The valid function codes are:

`EDT$K_OPEN_IN_OUT (4)`

Open the work file for both input and output. Neither the **record** nor **recordno** parameter is used.

`EDT$K_GET (5)`

Read a record. The **recordno** parameter is the number of the record to read. The **record** parameter gives the location where the record is to be stored.

`EDT$K_PUT (6)`

Write a record. The **recordno** parameter is the number of the record to write. The **record** parameter tells the location of the record to be written.

`EDT$K_CLOSE_DEL (7)`

Close the work file. After a successful close, the file is deleted. Neither the **record** nor **recordno** parameter is used.

recordno

This parameter is not used for open or close calls. For EDT$K_GET and EDT$K_PUT, this parameter contains the number of the record to be read or written. EDT always writes a record before reading that record.

record

This parameter is not used for open or close calls. For read operations, **record** contains the location of the record to be read. For write operations, **record** tells where the record is to be written. This parameter always refers to a 512-byte string during get and put calls.

XLATE

```
status.wlc.v = xlate(string.mt.dx)
```

As in the main calling sequence, the parameter must have the correct characteristics for access type, data type, passing mechanism, and parameter form.

The parameters are:

status

This parameter is a VAX/VMS status code. It is normally a success code. If **xlate** cannot process the passed string for some reason, it sets **status** to an error code. Returning an error code from **xlate** will abort the current key execution and display the appropriate error message.

string

On input, this parameter is a text string passed to XLATE. Normally **xlate** is used by defining a keypad mode key function to have XLATEstring^Z as its definition. The string is passed by the string parameter. The string descriptor will be one that can be handled by the RTL routine STR$COPY_DX.

On output, this parameter is a text string returned to EDT. The string is made up of nokeypad commands that EDT will execute.

D.11.2 XLATE

When you are using EDT after it has been called by a running program, you can issue the nokeypad XLATE command to pass a string back to the running program. The string is used by the program to activate a prespecified routine or pass information to the program.

XLATE passes the string to the entry point **xlate** in the calling program.

The syntax for XLATE is:

```
XLATEstring^Z
```

The **string** specifier is the text that EDT passes back to the calling program. CTRL/Z signals the end of the string. You can use the DELETE key to edit the string before pressing CTRL/Z.

After you press CTRL/Z, press RETURN to send the XLATE string to the calling program.

A brief outline of actions that involve XLATE is as follows. The program calls EDT. During your editing session, you use XLATE to pass a string to the program. The program uses that string to activate the specified process. Control returns to EDT. When you end the EDT session, the program resumes its run.

Since XLATE is a nokeypad command, you must either enter nokeypad mode to use it, or define a keypad mode function key to process the XLATE command. If you plan to use XLATE from keypad mode, you can include the XLATE string in the key definition or use a prompt so that you can type different strings during the EDT session. For example:

```
DEFINE KEY GOLD X AS "XLATEshow users^Z."

DEFINE KEY GOLD X AS "XLATE?'XLATE here: '^Z."
```

If you use the prompt definition, remember to press ENTER to send the string to EDT. The period at the end of the key definitions causes EDT to pass the string to the calling program.

XLATE strings can be used for a variety of purposes. The string can cause a routine in the calling program to return a stream of nokeypad commands. As soon as this stream is passed back to EDT, the nokeypad commands are performed.

The string can be saved for future use by the program routine. In this case, the program returns a null string to EDT, which EDT interprets as no-operation.

You can use the XLATE string to save data for future use and to have an EXT EXIT command returned by the program to EDT to end the EDT session.

The string can be passed to the user's XLATE routine in the program. Depending on the string that you supply, the system performs different operations.

This sample BASIC routine enables you to use DCL commands during your EDT session to do such things as print your directory, show various system settings, or list the current system users.

```
10 FUNCTION INTEGER XLATEROUT( CMD$ )
20 DECLARE INTEGER SUBPROC
30 CALL LIB$SPAWN('$ DEL XLATE.LOG;')
40 CALL LIB$SPAWN(CMD$,,'XLATE.LOG;',,,,SUBPROC)
50 CMD$ = 'EXT CLEAR XLATE;F=XLATE;INC XLATE.LOG;'
60 XLATEROUT = SUBPROC
70 FUNCTIONEND
```

You use the XLATE command to actually issue the DCL commands during your editing session. The following nokeypad command causes the calling program containing the previous routine to list your directory during your EDT session:

```
XLATEdirectory^Z
```

D.11.3 Callable EDT Example in VAX–11 BASIC

The following example uses the EDT-supplied routines EDT$FILEIO and EDT$WORKIO for the file and work system I/Os. The XLATE routine is called AXLATE. By passing 0 (zero) as the option word, you are specifying /NORECOVER, that is, normal command file processing will be used. The default /JOURNAL qualifier is in effect and will use the default journal file name.

The name of the file to be edited is FILE.BAS.

```
1     100       EXTERNAL INTEGER EDT$FILEIO
1     200       EXTERNAL INTEGER EDT$WORKIO
1     250       EXTERNAL INTEGER AXLATE
1     300       EXTERNAL INTEGER FUNCTION EDT$EDIT
1     400       DECLARE INTEGER RESULT
1     450       DIM INTEGER PASSFILE(1%)
1     460       DIM INTEGER PASSWORK(1%)
1     465       DIM INTEGER PASSXLATE(1%)
1     470       PASSFILE(0%) = LOC(EDT$FILEIO)
1     480       PASSWORK(0%) = LOC(EDT$WORKIO)
1     490       PASSXLATE(0%) = LOC(AXLATE)
1     500       RESULT = EDT$EDIT('FILE.BAS','','EDTINI','', 0%, &
                  PASSFILE(0%)BY REF, PASSWORK(0%)BY REF,
                  ,PASSXLATE(0%)BY REF)
1     600       IF (RESULT AND 1%) = 0%
2               THEN
2                   PRINT "SOMETHING WRONG"
3                   CALL LIB$STOP(RESULT BY VALUE)
1     900       PRINT "EVERYTHING O.K."
1    1000      END
```

The following example illustrates how to open a file and signal if the open fails. The symbol EDT$K_FAC_NO should be changed to the facility number of the caller of callable EDT.

```
BEGIN

!+
! Open the input file, signaling if the open fails.
!-

LOCAL
    OPEN_STATUS,
    CONNECT_STATUS;

OPEN_STATUS = $OPEN (FAB = INPUT1_FAB);

IF ( NOT .OPEN_STATUS)
THEN

!+
! The OPEN failed, signal "error opening for input", with the file
! name and RMS status code.
!-

    SIGNAL_STOP (
          SHR$_OPENIN + (EDT$K_FAC_NO*65536) + STS$K_SEVERE,
          1,
          INPUT1_FILE_NAME_DESC,
          .INPUT1_FAB [FAB$L_STS], .INPUT1_FAB [FAB$L_STV]);

CONNECT_STATUS = $CONNECT (RAB = INPUT1_RAB);

IF ( NOT .CONNECT_STATUS)
THEN

!+
! The CONNECT failed; signal similarly.
!-

    SIGNAL_STOP (
          SHR$_OPENIN + (EDT$K_FAC_NO*65536) + STS$K_SEVERE,
          1,
          INPUT1_FILE_NAME_DESC,
          .INPUT1_RAB [RAB$L_STS], .INPUT1_RAB [RAB$L_STV]);

END;
```

Appendix E
Using EDT with the RSTS/E Operating System

This appendix contains information about using EDT with the RSTS/E operating system. It describes procedures for starting an EDT session, the qualifiers that modify the EDIT command, EDT's use of files, control characters that should not be defined, special terminal settings, and chaining to EDT from a running program.

E.1 Calling Up EDT

Under the DCL keyboard monitor, the RSTS/E commands that call up EDT are:

```
$ EDIT [/QUALIFIER(s)]
File:     file-specification

$ EDIT [/QUALIFIER(s)] file-specification
```

In CCL, the RSTS commands that call up EDT are:

```
Ready

EDT
EDT> [[output-file] [,journal-file]=] input-file
        [,command-file] [/QUALIFIER(s)]

Ready

EDT [[output-file] [,journal-file]=] input-file
       [,command-file] [/QUALIFIER(s)]
```

Note: when you are working under RSTS/E DCL, you have the option of using the CCL **EDT** command with its qualifiers and specifiers.

In DCL, the prompt **File:** appears whenever you do not supply a file name on the EDIT command line. The prompt for you to enter the file specification in CCL is EDT>. The file specification must include at least the file name. No wildcard characters can be included. Use of the file type (for example .DAT) is optional. File specifications can include directory designations as appropriate, for example:

```
EDIT SY:[250,251]FILE.DAT
```

The default editor for RSTS/E systems is EDT. For DCL, you only need to type the EDIT command in order to call up EDT. However, if your system manager has designated another editor to be the default on your system, you must type EDIT /EDT to call up EDT.

E.2 RSTS/E Files with EDT

RSTS/E file names can have no more than six characters.

The examples in this manual generally include version numbers as part of the file specification. RSTS/E files do not have version numbers. When you end your session by typing the EDT EXIT command without an output file specification, EDT preserves the original version of your file, assigning the file type **.BAK** to it. The newly edited version has the file type you specified.

For example, suppose you first create a file called CHAP1.RNO. After you use EXIT to end your session, your directory contains the file CHAP1.RNO. Next, use EDT to edit that file. This time when you type EXIT, the original version of the file is renamed CHAP1.BAK and the newly edited text has the file name CHAP1.RNO.

If you use EDT to edit CHAP1.RNO again and then type EXIT, the first version is deleted from your directory. The second version is now called CHAP1.BAK and the third version is CHAP1.RNO.

If you need to have three or more versions in your directory at the same time, you must use a different name for some of them. You can use the RSTS/E system RENAME or COPY command to accomplish this. You can also use the /OUTPUT = output-file qualifier in your DCL EDT command line. For example:

```
EDIT /OUTPUT=CHAP1A.RNO CHAP1.RNO
```

Under CCL, a new output file name can be indicated this way:

```
EDT CHAP1A.RNO = CHAP1.RNO
```

Now three files exist in your directory:

```
CHAP1A.RNO
CHAP1.RNO
CHAP1.BAK
```

You can also assign a different name to the output file that contains the newly edited text by specifying the new name with EDT's EXIT command.

```
*EXIT CHAP1A.RNO
```

E.3 EDT Default File Attributes

EDT carries certain file attributes from the primary input file to the primary output file. If there is no primary input file, EDT uses certain defaults for these attributes. The WRITE and PRINT commands always create files with the default attributes. The attributes of a file read by the INCLUDE command have no effect on the attributes of the primary output file.

The file attributes that EDT carries have these default values:

Attribute	Default
Protection	same as the job default protection
Organization	sequential
Record Format	stream
Record Attribute	none

If you use the /FORMAT = VARIABLE qualifier, the default record format is variable, rather than stream and the record attribute is carriage return.

If you use EDT's /SEQUENCE qualifier with the EXIT or WRITE command, the Record Format and the Record Attribute change as follows:

Record Format	VFC (Variable with Fixed length Control) with a maximum length of 255 bytes
Record Attribute	carriage return

E.4 DCL Command Qualifiers for EDT

The RSTS/E operating system uses several qualifiers with the DCL EDIT /EDT command. Table E-1 lists all the EDIT /EDT qualifiers and their defaults. Explanations of the qualifiers appear after the table. The examples show the qualifiers typed directly after the EDIT /EDT command, before the input file specification. You can place the qualifiers anywhere on the command line after the initial EDIT /EDT command.

Table E.1 DCL EDIT Command Qualifiers

Command Qualifiers	Default
/[NO]COMMAND[= command-file]	/COMMAND = LB:EDTSYS.EDT
/[NO]CREATE	/CREATE
/FORMAT = STREAM /FORMAT = VARIABLE	same as input file
/[NO]JOURNAL[= journal-file]	/JOURNAL
/[NO]OUTPUT[= output-file]	/OUTPUT
/[NO]RO	/NORO
/[NO]RECOVER	/NORECOVER

/COMMAND=command-file
/NOCOMMAND

Determines whether EDT uses a startup command file or not. The /COMMAND qualifier is followed by an equal sign (=) and then the specification of the startup command file. **.EDT** is the default file type for startup command files.

```
EDIT /COMMAND=XEDTIN.EDT MEMO.DAT
```

If you do not include the /COMMAND = command-file qualifier, EDT processes the system-wide startup command file LB:EDTSYS.EDT. If this file does not exist, EDT looks for the file EDTINI.EDT in your default directory. If neither file exists and you have not used the /COMMAND qualifier, EDT begins your editing session in the default state.

When you do not want EDT to process either the system-wide startup command file or the EDTINI.EDT file in your default directory, use the /NOCOMMAND qualifier.

```
EDIT /NOCOMMAND MEMO.DAT
```

See Chapter 7 for information on creating and using startup command files.

/CREATE
/NOCREATE

Controls whether EDT creates a new file when the specified input file is not found. Normally, EDT creates a new file to match the input file specification if it cannot find that file name in the specified directory. When you use the /NOCREATE qualifier in the EDT command line and accidentally type a specification for a file that does not exist, EDT prints an error message and returns you to the system command level.

```
EDIT /NOCREATE LETTRX.RNO
```

/FORMAT=STREAM
/FORMAT=VARIABLE

Determines the format of the output file. Normally, you will be editing files in STREAM format. Use the /FORMAT = VARIABLE qualifier to create variable-length output files. (RMS-11 documentation gives information about stream and variable-length file formats.)

```
EDIT /FORMAT=VARIABLE PRGOUT.COB
```

/JOURNAL=journal-file
/NOJOURNAL

Determines whether EDT keeps a journal file during your EDT session. The default file name for the journal file is the same as the input file name. EDT uses **.JOU** as the default file type. The /JOURNAL qualifier enables you to use a different file specification for the journal file, for example:

```
EDIT /JOURNAL=SAVE.USE MEMO.DAT
```

If you are editing a file from another directory and want the journal file to be located in that directory, you must use the /JOURNAL qualifier with a file specification that includes a directory name. Otherwise, EDT creates the journal file in the current directory.

If you do not want EDT to keep a record of your editing session operations, use the /NOJOURNAL qualifier in the EDT command line.

```
EDIT /NOJOURNAL LETTER.DAT
```

Use the /RECOVER qualifier to have EDT process the commands contained in the journal file. Details on using EDT's journal facility appear in Chapters 2 and 7.

/OUTPUT=output-file
/NOOUTPUT

Determines whether EDT creates an output file at the end of your editing session. An equal sign (=) separates the /OUTPUT qualifier from the output file specification. The default file specification for the output file is the same as that of the input file. Use the /OUTPUT qualifier to have the MAIN buffer text put in a file with a different specification.

```
EDIT /OUTPUT=OUTMEM.DAT MEMO.DAT
```

You can include directory information as part of your output file specification to send output to another directory.

```
EDIT /OUTPUT=[250,251]OUTMEM.DAT MEMO.DAT

EDIT /OUTPUT=SY:[240,242]OUTMEM.DAT MEMO.DAT
```

The /NOOUTPUT qualifier suppresses the creation of an output file, but not the creation of a journal file. If you are testing some edits and are not sure you want an output file, you can use the /NOOUTPUT qualifier in the EDIT command line. A system interruption will not prevent you from recreating your editing session because a journal file is still being maintained. If you later decide that you want to keep your editing work, you can save your edited text in spite of the /NOOUTPUT qualifier by using the line mode WRITE command to put the text in an external file before you end the session.

```
EDIT /NOOUTPUT EMLIST.DAT
```

/RO
/NORO

Determines whether EDT keeps a journal file and creates an output file. With /NORO, the default, EDT maintains the journal file in case a system interruption occurs and creates an output file when it processes the line mode EXIT command. The /RO qualifier has the effect of using the /NOJOURNAL and /NOOUTPUT qualifiers.

```
EDIT /RO MEMO.DAT
```

Use the /RO qualifier when you are merely looking for things in a file. If you are reading files in a library directory, you can use the /RO qualifier to avoid the chance of a journal file being left in that directory. If you are working in a directory that has been saved with write protection, you must use the /RO qualifier with EDT in order to examine the file.

/RECOVER
/NORECOVER

Determines whether EDT uses the journal file to restore editing work. Normally, EDT starts a session without regard to any journal file that might be in the current directory. This is the default /NORECOVER state. When you use the /RECOVER qualifier, EDT reads the journal file and processes the commands it contains. If the journal file type is not **.JOU** and the file name is not the same as the input file name, you must include both the /JOURNAL=journal-file qualifier *and* the /RECOVER qualifier in the EDIT command line.

```
EDIT /RECOVER MEMO.DAT

EDIT /RECOVER /JOURNAL=SAVE.JOU MEMO.DAT
```

Because /NORECOVER is the default for EDT, you need not specify it in a command line. For more information on the EDT journal/recovery facility, see Chapters 2 and 7.

E.5 CCL Command Qualifiers and Specifiers for EDT

The CCL command syntax for EDT involves both qualifiers, such as /RECOVER and /RO, and file specifications, such as output file and command file. The syntax is:

```
EDT [[output-file] [,journal-file]=] input-file
    [,command-file] [/QUALIFIER(S)]
```

The CCL EDT command line can take four different file specifications:

input-file	command-file
output-file	journal-file

You must include the file specification of the input file each time you call up EDT; the remaining file specifications are optional. The input file specification is always separated from any preceding file specification by the equal sign (=) and from a following file specification by the comma (,).

Two files can be used at the start of your editing session and two can be saved in a directory when the session ends.

Starting Files –

Input File: The file specification for the file you want to edit or create. You must supply this information.

Command File: The file specification for the startup command file for EDT to process when you begin the editing session. **.EDT** is the default file type for EDT startup command files. The specified command file supersedes any other command file. If you do not specify a command file in the EDT command line, EDT uses the system-wide startup command file LB:EDTSYS.EDT. If no such file exists, EDT looks for a file called EDTINI.EDT in the default directory. If neither file exists and no command file specification is supplied, your editing session begins in EDT's default state.

Ending Files –

Output File: The file specification for the external file that EDT creates when you end your session with the EXIT command. This file contains the copy of the MAIN buffer text. When you do not specify an output file in the command line or with EDT's EXIT command, EDT creates an output file with the same file specification as the input file.

Journal File: The file specification for the journal file. EDT uses **.JOU** as the default file type for journal files. If you do not include a journal file in the EDT command line, EDT creates a journal file with the same file name as the input file, but with **.JOU** for the file type. See Chapters 2 and 7 for more information on journal files.

Examples:

```
>EDT INTRO.RNO
     Input file:  INTRO.RNO
     Output file:  INTRO.RNO      (original file becomes INTRO.BAK)
     Journal file:  INTRO.JOU
     Command file, if one exists: LB:EDTSYS.EDT or EDTINI.EDT

>EDT CHAPT1.RNO= INTRO.RNO
     Input file:  INTRO.RNO
     Output file:  CHAPT1.RNO
     Journal file:  INTRO.JOU
     Command file, if one exists:  LB:EDTSYS.EDT or EDTINI.EDT

>EDT CHAPT1.RNO ,JOU= INTRO.RNO
     Input file:  INTRO.RNO
     Output file:  CHAPT1.RNO
     Journal file:  JOU.JOU
     Command file, if one exists:  LB:EDTSYS.EDT or EDTINI.EDT

>EDT INTRO.RNO ,DIFINI
     Input file:  INTRO.RNO
     Output file:  INTRO.RNO      (original file becomes INTRO.BAK)
     Journal file:  INTRO.JOU
     Command file:  DIFINI.EDT

>EDT CHAPT1.RNO ,JOU= INTRO.RNO ,DIFINI
     Input file:  INTRO.RNO
     Output file:  CHAPT1.RNO
     Journal file:  JOU.JOU
     Command file:  DIFINI.EDT
```

When you want EDT to use no startup command file, or no output file, or no journal file, include the position marker for that file, but do not supply any file specification.

```
>EDT CHAPT1.RNO ,= INTRO.RNO ,
     Input file:  INTRO.RNO
     Output file:  CHAPT1.RNO
     Journal file:  none
     Command file:  none

>EDT ,JOU= INTRO.RNO
     Input file:  INTRO.RNO
     Output file:  none
     Journal file:  JOU.JOU
     Command file, if one exists:  LB:EDTSYS.EDT or EDTINI.EDT
```

If you want to have the default output file produced but specify a different journal file, you must use the /NORO qualifier at the end of the command line.

```
>EDT ,JOU= INTRO.RNO /NORO
     Input file:  INTRO.RNO
     Output file:  INTRO.RNO
     Journal file:  JOU.JOU
     Command file, if one exists:LB:EDTSYS.EDT or EDTINI.EDT
```

The qualifiers available with the CCL command line are summarized in Table E-2.

Table E-2: CCL EDIT Command Qualifiers

Command Qualifiers	Default
/[NO]CHAIN[="chain parameter"]	/NOCHAIN
/[NO]CCL[="CCL command"]	/NOCCL
/NOCOMMAND	
/[NO]CREATE	/CREATE
/NOJOURNAL	
/NOOUTPUT	
/[NO]RECOVER	/NORECOVER
/[NO]RO (read only)	/NORO
/[NO]STREAM	same as input file
/[NO]VARIABLE	same as input file

You can use the minus sign (–) in place of **NO** for the negative form of the qualifier.

Most of the CCL qualifiers have the same functions as DCL's EDIT /EDT qualifiers. Information on /STREAM and /VARIABLE appears under DCL's /FORMAT qualifier. /CHAIN and /CCL are described here.

```
/CHAIN="file-specification[;number-parameter[;Core-Common-parameter]]"
/NOCHAIN
```

Enables you to chain from EDT to another program. You must surround the chain parameter with delimiters, which can be any nonalphanumeric character that does not appear in the string. For more information on chaining from EDT, see the section at the end of this appendix.

```
/CCL="CCL command"
/NOCCL
```

Enables you to chain from EDT to another program. The RSTS/E system uses the CCL command string to set both CORE COMMON and the number parameter for chaining from EDT. See the section on chaining from EDT at the end of this appendix.

Although EDT accepts the /COMMAND, /JOURNAL, and /OUTPUT qualifiers in the CCL EDT command line, you cannot use a file specification with these qualifiers; thus they have no effect on EDT. However, the NO forms (/NOCOMMAND, /NOJOURNAL, and /NOOUTPUT) cause EDT to use no command file or to create no journal file or output file.

Here are some examples of CCL command lines with qualifiers:

```
EDT /NOCOMMAND LETTER.DAT

EDT /RECOVER ,SAVE.JOU=LETTER.DAT

EDT /RO LB:SYSLIB.DAT
```

EDT scans the CCL command line from left to right. If you use the /NOJOURNAL qualifier and then include a journal file name, EDT will create a journal file with that name. Conversely, if you include a journal file name and then add the /NOJOURNAL qualifier after the input file name, EDT will not create a journal file for that editing session.

E.6 Control Characters That Cannot be Defined

When you are defining keys in EDT on a RSTS/E system, there are several control key sequences that cannot be used by EDT. These control keys already have system functions assigned to them. EDT allows you to define these keys, but when you try to use them, the operating system performs its control function and never passes the control key code to EDT.

For example, suppose you assign the definition (–W). to CTRL/S. Later you press CTRL/S to move the cursor back one word. Instead you find that the terminal output has been suspended, the RSTS/E function for CTRL/S. (CTRL/Q restores your terminal to normal operation after CTRL/S has been pressed.) When you resume normal operation, you see that pressing CTRL/S had no effect on the cursor position. Other control key sequences might have more serious consequences for your editing session.

These are the control key sequences that you should not define:

CTRL/C
CTRL/O
CTRL/P (from the console terminal)
CTRL/Q
CTRL/S
CTRL/X

E.7 Terminal Settings with EDT under the RSTS/E Operating System

Before using EDT under RSTS/E DCL, be sure that your terminal has been set up using DCL's SET TERMINAL command. Do not use the /HOSTSYNC qualifier with the /VT52 parameter since EDT uses the absence of /HOSTSYNC to distinguish a VT52 from a VT100.

VT100-type terminals without AVO (advanced video option) and VT52 terminals can only accommodate 80 characters per line. EDT gets its information about screen width directly from the RSTS/E terminal characteristics. If you have a screen width setting of over 80 characters in your EDT startup command file and try to use EDT on a VT100-type terminal without AVO, or a VT52 terminal, EDT loses track of lines and characters. You must reset the screen width to 80 either using the system SET WIDTH 80 command or EDT's SET SCREEN 80 command.

When you finish your editing session, EDT always sets VT100-type terminals back to either the 80 or 132 screen width depending on the width that your operating system has set for your terminal.

When you call up EDT to edit a file, EDT uses information from the RSTS/E system to determine what type of terminal you are using. This information separates terminals into three categories – VT100, VT52, and hardcopy.

VT100
If the terminal requires the RSTS/E XON switch, EDT assumes that it uses VT100 escape sequences, has scrolling regions, and has autorepeat suppression. EDT also assumes that the terminal does not have eightbit graphics or internal editing features.

VT52

If the terminal does not use the RSTS/E XON switch, EDT assumes that it uses VT52 escape sequences, has no scrolling regions, and has no autorepeat suppression. EDT also assumes that the terminal does not have eightbit graphics or internal editing features.

Hardcopy

If the RSTS/E system does not recognize the terminal as a scope or if the terminal has a maximum screen width of less than 80 characters per line, EDT assumes that it is a hardcopy terminal.

If your terminal has eightbit graphics and/or internal editing features, use EDT's SET TERMINAL EIGHTBIT and/or SET TERMINAL EDIT commands to have EDT make use of these additional features.

If your terminal has a special feature, you can use the appropriate EDT SET command to suppress EDT's use of the feature. However, if your terminal does not have a feature, you cannot use the SET TERMINAL command to add that capability. If you try to do this, EDT becomes confused and problems will occur in your editing session.

See Appendix C for more information on terminal types and characteristics.

E.8 The Tentative File with EXIT, PRINT, and WRITE

EDT creates a tentative file when executing an EXIT, PRINT, or WRITE command. The purpose of the tentative file is to make sure that EDT does not add a new file to a directory, that contains only part of the text you wanted it to have, without indicating its partial nature. When the file has been written successfully, EDT converts it to a normal file.

E.9 Information on EDT HELP Files

Different operating systems use different methods to access data in EDT's HELP file. When you create a new HELP file or modify the existing one, you can improve user access to the material by arranging the text in the order that your system uses.

In the default EDT HELP file, the keypad editing material is located just after the HELP topic. The remaining topics are arranged in alphabetical order. Subtopics are also listed alphabetically under their respective topics. This organization is designed both for systems that access information alphabetically as well as those that access information sequentially.

On RSTS/E systems, EDT accesses the HELP file sequentially. Thus, you can increase efficiency by arranging the HELP text in order of use. If your system has few screen terminals connected to it, you might want to place the keypad information towards the end of the HELP file. If many people are using the nokeypad information to create keypad definitions, you might place that text closer to the beginning of the file. See Chapter 7 for more information on creating and modifying EDT HELP files.

E.10 Chainable EDT

Chainable EDT enables you to access EDT from a running program and have EDT chain back to the original program or some other specified program. The program does the following:

1. Creates an EDT command line.
2. Calls up the EDT editor.

3. Allows a user to perform editing operations interactively at a terminal.
4. Either returns to the original program or continues on to another program.

You use the /CHAIN qualifier or the /CCL qualifier to establish the link from EDT to the program. The /CHAIN qualifier syntax is:

```
/CHAIN="file-specification[;number-parameter[;Core-Common-parameter]]"
```

The entire string after the equal sign must be enclosed by delimiters, which can be any nonalphanumeric character that does not appear in the string.

File-specification is any valid RSTS/E file notation (device:[ppn]file name.file-type). This file is the program to which EDT will return after completing its operations. The program can be either the one that invokes the chain to EDT or a different one.

Number-parameter is a decimal number from 0 to 32767 that generally specifies the BASIC program line number. When EDT has completed its operation, it returns to that line number in the specified program.

Core-Common-parameter is an ASCII string that is placed in CORE COMMON starting at location 462(8).

The /CCL qualifier can be used to set both the **number-parameter** and the **Core-Common-parameter** .

```
/CCL="CCL command"
```

The entire string after the equal sign must be enclosed by delimiters, which can be any nonalphanumeric character that does not appear in the string.

You can use a valid RSTS/E CCL command string with the /CCL qualifier to set the two parameters.

E.10.1 Requirements for the Program Chaining to EDT

The program that chains to EDT must do two things:

1. Place in CORE COMMON starting at 460(8) a one-byte count, that contains the length of the command line to be passed. This count must be followed by an EDT command line that has either the /CCL qualifier or the /CHAIN qualifier containing the necessary parameter information.

```
EDIT MYPROG /CHAIN="[50,50]MAIL.TSK;0;XYZ"

EDT MYPROG /CCL="MAIL XYZ"
```

Alternatively, a BASIC program can execute the statement:

```
Q$=SYS(CHR$(8%)+'EDT MYPROG /CCL="MAIL XYZ"')
```

2. Put a binary **30000.** in the parameter word of the FIRQB, and do a **.RUN** monitor directive on the executable task image of EDT – generally **[1,2]EDT.TSK** .

   Alternatively, a BASIC program can execute the statement:

```
CHAIN "[1,2]EDT.TSK" LINE 30000%
```

When EDT's EXIT command is issued, EDT checks to see whether the /CHAIN or /CCL qualifier is in the EDT command line.

E.10.2 Using the /CHAIN Qualifier

If the /CHAIN qualifier is present, EDT attempts to issue the **.RUN** monitor directive on the program (specified in the /CHAIN qualifier's file specification) after filling in CORE COMMON and the parameter word.

CORE COMMON contains three pieces of data:

First byte:	Indicates the number of remaining bytes in CORE COMMON. This number ranges from 1 to 127. If it is 1, it means that only the status byte (the second byte) is present. Any other data in CORE COMMON will not be erased.
Second byte:	Indicates the status of the operation. Binary 255 means success; binary 0 means failure.
Remaining bytes:	Contain the data supplied in the **Core-Common parameter** string. The maximum length for this string is 126 characters. This string is placed in CORE COMMON immediately after the status byte.

If the optional number parameter is present in the command line, it is converted from ASCII to binary and placed in the parameter word in the FIRQB. When the **.RUN** directive is executed, this data is passed to the invoked run-time system as the parameter word. When you are chaining to a BASIC program, this parameter is the line number in the program where you want processing to resume after you have finished the EDT work. If you supply neither the number parameter nor the Core-Common parameter, the command line passes a binary zero to the program. If you do not need to supply a number parameter, but want to include the Core-Common parameter, you must put a zero in the numeric parameter position on the command line.

Note that EDT will set the SIGN bit in the number parameter and raise privileges only when it was entered with a parameter of 30000 on the **.RUN** or chain. This enables you to pass back temporary privileges to the chaining program. (See the section on privileged programs for more information.)

E.10.3 Using the /CCL Qualifier

If the /CCL qualifier is present in the command line, EDT attempts to issue the **.CCL** monitor directive on the program specified in the CCL command string. If the **.CCL** directive succeeds, the specified program runs. The RSTS/E system places the CCL command string in CORE COMMON. No status byte is returned in CORE COMMON.

Note that the /CCL qualifier cannot be used to raise privileges. (See the section on privileged programs for more information.)

E.10.4 Failure of Monitor Directives

The /CHAIN qualifier invokes the **.RUN** monitor directive; the /CCL qualifier invokes the **.CCL** monitor directive. If either of these directives fails, EDT simply returns to the job default keyboard monitor. No warning message or indication is given.

E.10.5 Privileged Programs

Suppose you want to chain from a privileged program to EDT and then retain those same privileges when you use the number parameter (for example, a line number in a BASIC program) to return to the program or move to another program. Two things must be in effect:

- The number parameter passed in the FIRQB or the line number used in the BASIC CHAIN statement must include a SIGN bit setting. The SIGN bit signals the RSTS/E system to pass EDT temporary privileges equivalent to those of the privileged program.
- EDT must have the **executable/privileged** protection code (normally 232), so that the temporary privileges can be passed back to the program after EDT's EXIT command is given.

Since EDT drops the temporary privileges when it takes control, it must be able to regain them before issuing the **.RUN** monitor directive with the /CHAIN qualifier.

If the /CCL qualifier is used, the normal RSTS/E CCL mechanisms are in effect. In order for the invoked program to be privileged, it must have a **privileged** protection code either with no CCL command word or with the PRIV keyword included in the CCL command word.

Appendix F
Using EDT with the RSX-11M and RSX-11M-PLUS Operating Systems

This appendix contains information about using EDT with the RSX-11M and RSX-11M-PLUS operating systems. The sections describe the procedures for starting an EDT session, the qualifiers that modify the EDIT /EDT command, EDT's use of files, the control characters that should not be defined, and special terminal settings. The final section includes a sample utility that you can use to call EDT from a running program.

F.1 Calling Up EDT

The syntax for the RSX-11M/M-PLUS command line that calls up EDT is:

```
DCL>EDIT /EDT [/qualifier(s)] input-file

DCL>EDIT /EDT [/qualifier(s)]
File? input-file

MCR>EDT [[output-file][,journal-file]=] input-file[,command-file]
                [/qualifier(s)]

MCR>EDT [/qualifier(s)]
EDT> [[output-file] [,journal-file]=] input-file [,command-file]
```

DCL uses the **File?** prompt to ask for the input file specification if you do not supply one in your EDIT/EDT command line. MCR uses the prompt **EDT>** to request that information. File specifications can include directory information as appropriate.

```
EDIT /EDT [131,22]FILE.DAT
```

F.2 DCL – EDT Command Line Qualifiers and Specifiers

The RSX-11M/M-PLUS systems use several qualifiers with the EDIT /EDT command. Table F-1 lists the qualifiers and their defaults. Explanations of the qualifiers appear after the table. The examples show the qualifiers typed directly after the EDT command and before the input file specification. You can place the qualifiers anywhere in the command line after /EDT. The minus sign (–) conveys the same meaning as the initial letters **NO** for RSX-11M/M-PLUS qualifiers. When a file specification follows a qualifier, the file specification must be preceded either by a colon (:) or an equal sign (=).

Table F-1: RSX EDIT /EDT DCL Command Qualifiers

Command Qualifier	Default
/[NO]COMMAND[:command-file]	/COMMAND:LB:[1,2]EDTSYS.EDT
/[NO]CREATE	/CREATE
/[NO]JOURNAL[:journal-file]	/JOURNAL:file-spec.JOU
/[NO]OUTPUT[:output-file]	/OUTPUT:file-spec.type;n + 1
/[NO]RO	/NORO
/[NO]RECOVER	/NORECOVER

/COMMAND:command-file
/-COMMAND
/NOCOMMAND

Determines whether EDT uses a startup command file or not. The /COMMAND qualifier is followed by a colon (:) and then the specification of the startup command file. **.EDT** is the default file type for startup command files.

```
EDIT /EDT /COMMAND:NEWEDTINI.EDT BASICPROG.BAS
```

If you do not include the /COMMAND:command-file qualifier, EDT processes the system-wide startup command file LB:[1,2]EDTSYS.EDT. If this file does not exist, EDT looks for the file EDTINI.EDT in your default directory. If neither file exists and you have not used the /COMMAND qualifier, EDT begins your editing session in the default state.

When you do not want EDT to process either the system-wide startup command file or the EDTINI.EDT file in your default directory, use either the /–COMMAND or /NOCOMMAND qualifier.

```
EDIT /EDT /-COMMAND NEWFILE.DAT

EDIT /EDT /NOCOMMAND NEWFILE.DAT
```

See Chapter 7 for information on creating and using startup command files.

/CREATE
/-CREATE
/NOCREATE

Controls whether EDT creates a new file when the specified input file is not found. Normally, EDT creates a new file to match the input file specification if it cannot find that file name in the specified directory. When you use the /NOCREATE qualifier in the EDT command line and accidentally type a specification for a file that does not exist, EDT prints an error message and returns you to the system command level.

```
EDIT /EDT /NOCREATE LETTERX.RNO
```

/JOURNAL:journal-file
/-JOURNAL
/NOJOURNAL

Determines whether EDT keeps a journal file during your editing session. The default file name for the journal file is the same as the input file name. EDT uses **.JOU** as the default file type. The /JOURNAL qualifier enables you to use a different file specification for the journal file, for example:

```
EDIT /EDT /JOURNAL:EDITS LIST.DAT
```

If you are editing a file from another directory and want the journal file to be located in that directory, you must use the /JOURNAL qualifier with a file specification that includes the directory name. Otherwise, EDT creates the journal file in the current directory.

You cannot use the /JOURNAL qualifier with /RO when you are working in a write-protected directory.

If you do not want EDT to keep a record of your editing session operations, use the /–JOURNAL or /NOJOURNAL qualifier in the EDT command line.

```
EDIT /EDT /-JOURNAL ADDRESS.DAT

EDIT /EDT /NOJOURNAL ADDRESS.DAT
```

You must use the /RECOVER qualifier to have EDT process the commands contained in the journal file. Details on using EDT's journal facility appear in Chapters 2 and 7.

/OUTPUT:output-file
/-OUTPUT
/NOOUTPUT

Determines whether EDT creates an output file at the end of your editing session. A colon (:) separates the /OUTPUT qualifier from the output file specification. The default file specification for the output file is the same as that of the input file but with a version number one greater than the input version. (In the case of new files, the output version number is always 1.) Use the /OUTPUT qualifier to have the MAIN buffer text put in a file with a different specification.

```
EDIT /EDT /OUTPUT:PARTYLIST.DAT EMPLOYEES.DAT
```

You can include directory information as part of your output file specification to send output to another directory.

```
EDIT /EDT /OUTPUT:[263,272]PARTYLIST.DAT EMPLOYEES.DAT

EDIT /EDT /OUTPUT:MT3:[145,34]PARTYLIST.DAT EMPLOYEES.DAT
```

The /NOOUTPUT qualifier suppresses the creation of an output file, but not the creation of a journal file. If you are testing some edits and are not sure you want an output file, you can use the /–OUTPUT or /NOOUTPUT qualifier in the EDT command line. A system interruption will not prevent you from recreating your editing session because a journal file is

still being maintained. If you later decide that you want to keep your editing work, you can save your edited text in spite of the /NOOUTPUT qualifier by using the line mode WRITE command to put the text in an external file before you end the session.

```
EDIT /EDT /NOOUTPUT EMPLOYEES.DAT
```

/RO
/-RO
/NORO

Determines whether EDT keeps a journal file and creates an output file. With /NORO, the default, EDT maintains the journal file in case a system interruption occurs and creates an output file when it processes the line mode EXIT command. The /RO qualifier has the effect of using the /NOJOURNAL and /NOOUTPUT qualifiers.

```
EDIT /EDT /RO LB:[1,2]EDTSYS.EDT
```

Use the /RO qualifier when you are merely looking for things in a file. If you are reading files in a library directory, you can use the /RO qualifier to avoid the chance of a journal file being left in that directory. If you are looking at a file in a directory that has been saved with write protection, you must use the /RO qualifier with EDT in order to examine the file.

/RECOVER
/-RECOVER
/NORECOVER

Determines whether EDT uses the journal file to restore editing work. Normally, EDT starts a session without regard to any journal file that might be in the current directory. This is the default /NORECOVER state. When you use the /RECOVER qualifier, EDT reads the journal file and processes whatever commands it contains. If the journal file type is not **.JOU** and the file name is not the same as the input file name, you must include both the /JOURNAL:journal-file qualifier *and* the /RECOVER qualifier in the EDT command line.

```
EDIT /EDT /RECOVER LETTER1.RNO

EDIT /EDT /RECOVER /JOURNAL:LETTEMP LETTER1.RNO

EDIT /EDT /RECOVER /JOURNAL:FORMLET.TEM LETTER1.RNO
```

Because /NORECOVER is the default for EDT, you need not specify it in a command line. For more information on the EDT journal/recovery facility, see Chapters 2 and 7.

F.3 MCR – EDT Command Line Qualifiers and Specifiers

The MCR command syntax for EDT involves both qualifiers, such as /RECOVER and /RO, and file specifications, such as output file and command file. The syntax is:

```
EDT [/qualifier(s)] [output-file[,journal-file]=] input-file
                    [,command-file]
```

The MCR EDT command line can take four different file specifications:

input-file	command-file
output-file	journal-file

You must include the file specification of the input file each time you call up EDT; the remaining file specifications are optional. The input file specification is always separated from any preceding file specification by the equal sign (=) and from a following file specification by the comma (,).

Two files can be used at the start of your editing session and two can be saved in a directory when the session ends.

Starting Files –

Input File: The file specification for the file you want to edit or create. You must supply this information.

Command File: The file specification for the startup command file for EDT to process when you begin the editing session. The default file type for EDT startup command files is **.EDT** . The specified command file supersedes any other command file. If you do not specify a command file in the EDT command line, EDT uses the system-wide startup command file LB:[1,2]EDTSYS.EDT. If no such file exists EDT looks for a file called EDTINI.EDT in the default directory. If neither file exists and no command file specification is supplied, your editing session begins in EDT's default state.

Ending Files –

Output File: The file specification for the external file that EDT creates when you end your session with the EXIT command. This file contains the copy of the MAIN buffer text. When you do not specify an output file in the command line or with EDT's EXIT command, EDT creates an output file with the same file specification as the input file, but increases the version number by 1.

Journal File: The file specification for the journal file. EDT uses **.JOU** as the default file type for journal files. If you do not include a journal file name in the EDT command line, EDT creates a journal file with the same file name as the input file, but with **.JOU** for the file type. See Chapters 2 and 7 for more information on journal files.

Examples:

```
>EDT INTRO.RNO
     Input file:  INTRO.RNO
     Output file:  INTRO.RNO;n+1
     Journal file:  INTRO.JOU
     Command file, if one exists:  LB:[1,2]EDTSYS.EDT or EDTINI.EDT

>EDT CHAPTER1.RNO= INTRO.RNO
     Input file:  INTRO.RNO
     Output file:  CHAPTER1.RNO
     Journal file:  INTRO.JOU
     Command file, if one exists:  LB:[1,2]EDTSYS.EDT or EDTINI.EDT
```

```
>EDT CHAPTER1.RNO ,JOU= INTRO.RNO
     Input file:  INTRO.RNO
     Output file:  CHAPTER1.RNO
     Journal file:  JOU.JOU
     Command file, if one exists:  LB:[1,2]EDTSYS.EDT or EDTINI.EDT

>EDT INTRO.RNO ,DIFEDTINI
     Input file:  INTRO.RNO
     Output file:  INTRO.RNO;n+1
     Journal file:  INTRO.JOU
     Command file: DIFEDTINI.EDT

>EDT CHAPTER1.RNO ,JOU= INTRO.RNO ,DIFEDTINI
     Input file:  INTRO.RNO
     Output file:  CHAPTER1.RNO
     Journal file:  JOU.JOU
     Command file:  DIFEDTINI.EDT
```

When you want EDT to use no startup command file, or no output file, or no journal file, include the position marker for that file, but do not supply any file specification.

```
>EDT CHAPTER1.RNO ,= INTRO.RNO ,
     Input file:  INTRO.RNO
     Output file:  CHAPTER1.RNO
     Journal file:  none
     Command file:  none

>EDT ,JOU= INTRO.RNO
     Input file:  INTRO.RNO
     Output file:  none
     Journal file:  JOU.JOU
     Command file, if one exists:  LB:[1,2]EDTSYS.EDT or EDTINI.EDT
```

If you want to have the default output file produced but specify a different journal file, you must use the /NORO qualifier at the end of the command line.

```
>EDT ,JOU= INTRO.RNO /NORO
     Input file:  INTRO.RNO
     Output file:  INTRO.RNO
     Journal file:  JOU.JOU
     Command file, if one exists:  LB:EDTSYS.EDT or EDTINI.EDT
```

The qualifiers available with the MCR command line are summarized in Table F-2.

Table F-2 MCR EDIT Command Qualifiers

Command Qualifiers	Default
/NOCOMMAND	
/[NO]CREATE	/CREATE
/NOJOURNAL	
/NOOUTPUT	
/[NO]RECOVER	/NORECOVER
/[NO]RO	/NORO

You can use the minus sign (–) in place of *NO* for the negative form of the qualifier.

The MCR qualifiers have the same functions as the DCL EDIT /EDT qualifiers. Although EDT accepts /COMMAND, /JOURNAL, and /OUTPUT in the command line, you cannot use a file specification with these qualifiers, so they have no effect on EDT. However, the NO forms (/NOCOMMAND, /NOJOURNAL, and /NOOUTPUT) cause EDT to use no command file, or to create no journal file, or no output file.

Here are some examples of MCR command lines with qualifiers:

```
EDT LETTER.DAT /NOCREATE

EDT ,SAVE.JOU=LETTER.DAT /RECOVER

EDT PROGRAM10.FOR /RO
```

EDT scans the MCR command line from left to right. If you use the /NOJOURNAL qualifier and then include a journal file name, EDT will create a journal file with that name. Conversely, if you include a journal file name and then add the /NOJOURNAL qualifier after the input file name, EDT will not create a journal file for that editing session.

F.4 EDT Default File Attributes

EDT carries certain file attributes from the primary input file to the primary output file. If there is no primary input file, EDT uses certain defaults for these attributes. The WRITE and PRINT commands always create files with the default attributes. The attributes of a file read by the INCLUDE command have no effect on the attributes of the primary output file.

The file attributes that EDT carries have these default values:

Attribute	Default
Protection	default protection code assigned to the output disk
Organization	sequential
Record Format	variable length
Record Attribute	carriage return
If you use the EDT /SEQUENCE qualifier with the EXIT or WRITE command the Record Format changes.	
Record Format	VFC (Variable with Fixed length Control)

Note that the output file is always noncontiguous.

F.5 Default Output Files

When you use EDT's EXIT command to end your editing session, EDT normally creates a new output file containing the text from your session's MAIN buffer. The default file specification is always the same as the input file specification you supplied in your EDT command line; the default version number is always one greater than the highest version number of the input file.

When you use EDT to create a new file, no version of that file exists, until you use EDT's EXIT command to complete your session. The new output file is assigned a version number of 1.

The version numbers in RSX-11M/M-PLUS systems are octal and range from 1 through 177777. Thus, if you are editing version 7 of a file, the version number of the output file will be 10 (assuming no higher version exists).

Input File				Output File
[new file]	→	EDT	→	FILE.FTN;1
FILE.FTN;1	→	EDT	→	FILE.FTN;2
FILE.FTN;7	→	EDT	→	FILE.FTN;10

When you supply a different output file specification with the /OUTPUT qualifier or EDT's EXIT command, EDT names the output file using that information rather than the default convention. No output file is created when you use the /RO or /NOOUTPUT qualifier, or when you use EDT's QUIT command.

F.6 Control Characters That Cannot be Defined

When you are defining keys in EDT on an RSX-11M/M-PLUS system, there are several control key sequences that cannot be used by EDT. These control keys already have system functions assigned to them. EDT allows you to define these keys, but when you try to use them, the operating system performs its control function and never passes the control key code to EDT.

For example, suppose you assign the definition **(–W).** to CTRL/S. Later you press CTRL/S to move the cursor back one word. Instead you find that the terminal output has been suspended, the RSX function for CTRL/S. (CTRL/Q restores your terminal to normal operation after CTRL/S has been pressed.) When you resume normal operation, you see that pressing CTRL/S had no effect on the cursor position. Other control key sequences might have more serious consequences for your editing session.

These are the control key sequences that you should not define:

CTRL/C
CTRL/O
CTRL/P (from the console terminal)
CTRL/Q
CTRL/S
CTRL/X

F.7 Using Terminals with EDT and the RSX-11M/M-PLUS Systems

When you call up EDT to edit a file, EDT collects certain information from your RSX system about the type of terminal you are using. This information includes data on the screen width, the number of bits that are used to represent a character, the existence of automatic scrolling regions, and the presence of internal editing features in the terminal.

Use EDT's SET SCREEN command to change the number of characters displayed on a line. Hardcopy terminals that take paper 15 inches wide can use a SET SCREEN width of 1 to 132 characters. VT100-type terminals with AVO (advanced video option) can use either SET SCREEN 80 or SET SCREEN 132. The only SET SCREEN width that you can use with VT52 terminals and VT100-type terminals without AVO is 80.

You can also use a system command to correct screen width settings. The commands are:

```
DCL> SET TERMINAL /WIDTH:80

MCR> SET /BUF=TI:80.
```

The period at the end of the MCR command line indicates to the system that 80 is a decimal number. If you omit the period, the system tries to interpret the value as an octal number and will reject it because 8 is not a valid octal digit.

When you finish your editing session, EDT always sets VT100-type terminals back to either the 80 or 132 screen width depending on the width that your operating system has set for your terminal.

EDT determines the screen width, the presence of internal editing features, and ability to handle eightbit graphics from the terminal characteristics. EDT uses the information it gets from the operating system to separate terminals into the following categories:

Category 1: VT100 or VT125
VT100 escape sequences, scrolling regions, autorepeat, possible internal editing, possible eightbit graphics

Category 2: VT100-type other than VT100 or VT125
VT100 escape sequences, scrolling regions, possible internal editing, autorepeat, possible eightbit graphics

Category 3: VT100 subset
VT100 escape sequences, no scrolling regions, no autorepeat, possible internal editing, possible eightbit graphics

Category 4: VT52
VT52 escape sequences, no scrolling regions, no autorepeat, no internal editing, no eightbit graphics

Category 5: all others
hardcopy, no scrolling regions, no autorepeat, no internal editing, no eightbit graphics

If your terminal has a special feature, you can use an appropriate EDT SET command to suppress EDT's use of the feature. However, if your terminal does not have a feature, you cannot use the SET TERMINAL command to add that capability. If you try to do this, EDT becomes confused and problems will occur in your editing session.

F.8 The Temporary File with EXIT, PRINT, and WRITE

EDT creates a temporary file whenever a system interruption occurs in the middle of an EXIT, PRINT, or WRITE command. The temporary file has the same file name as the file you are editing, but uses **.TMP** as the file type. The **.TMP** file type signals you that the temporary file probably does not contain all the text that you wanted to have in the external file.

In most instances, you will want to delete the **.TMP** file and rerun your EDT session, using the /RECOVER qualifier to process the journal file. The purpose of the **.TMP** file type is to alert you that the file contains only part of the text you wanted it to have.

F.9 Information on EDT HELP Files

Different operating systems use different methods to access data in EDT's HELP file. When you create a new HELP file or modify the existing one, you can improve user access to the material by arranging the text in the order that your system uses.

In the default EDT HELP file, the keypad editing information is located just after the HELP topic. The remaining topics are arranged in alphabetical order. Subtopics are also listed alphabetically under their respective topics. This organization is designed both for systems that access information alphabetically as well as those that access information sequentially.

Since EDT on RSX-11M/M-PLUS systems accesses its HELP file sequentially, you can increase efficiency by arranging the HELP text in order of use. If your system has few screen terminals connected to it, you might want to place the keypad information towards the end of the HELP file. If many people are using the nokeypad information to create keypad definitions, you might place that text closer to the beginning of the file. See Chapter 7 for more information on creating and modifying EDT HELP files.

F.10 Running EDT on an RSX-11M Operating System

To run EDT on an RSX-11M system, you must select the full-duplex terminal driver when you sysgen your RSX-11M system. Be sure to include all of the following options:

1. Unsolicited input character AST
2. Get multiple characteristics (SF.GMC)
3. Set multiple characteristics (SF.SMC)
4. Read after prompt (IO.RPR)
5. Get terminal support (IO.GTS)

If you do not include these options, EDT will treat your terminal as a hardcopy terminal.

These options are all mandatory on RSX-11M-PLUS systems and therefore need not be specified.

F.11 The Location of the Work File

For RSX-11M/M-PLUS systems, EDT opens its work file on LUN 1. You can move EDT's work file to another disk if you need to. After installing EDT, issue this MCR command:

```
REA ...EDT 1 xx0:
```

"xx0" is the name of an initialized FILES-11 disk mounted using the /SYS qualifier. The disk does not need to have any UFDs on it, except those created by the INI command, since EDT's work file does not have a name and therefore does not participate in the directory structure. Once you have done this, all EDT users will access the same disk for EDT's work file, regardless of which disk their files are on.

F.12 Calling EDT from a Program on RSX-11M/M-PLUS Systems

The SPAWN facility enables you to access EDT from a running program. The CALL statement calls the SPAWN routine, which in turn accesses EDT. The following FORTRAN sample calls on EDT to edit the file TEXT.DAT.

```
CALL SPAWN (RAD50('...EDT'),,,1,,,,'EDT TEXT.DAT',12,,,IDS)
CALL STLOR (1)
TYPE 10
FORMAT (' EDIT OF TEXT.DAT DONE')
END
```

Here is a fully worked example using RSX MACRO-11 that shows how to call EDT from a program. You need to create three files:

EDITOR.DEF;1	Editor and default extension definitions
EDIT.CMD	Task build command file
EDIT.MAC	MACRO-11 source program

Here are the contents for these three files:

File 1: `[EDITOR.DEF;1]`

```
EDIT.MAI
.MAC
ED3
```

File 2: `[EDIT.CMD]`

```
;
;     Note, /MU is only supported on M-PLUS
;
EDIT/MU/CP/-FP,EDIT/-SP=EDIT
/
UNITS = 3
ASG = TI:1
ASG = SY:2
PAR = GEN
PRI = 100
TASK = ...EDI
;
; FCS library options:
;
;     supervisor mode library:
;
; SUPLIB=FCSFSL:SV
;
;     or resident library:
;
; RESLIB=LB:[1,1]FCSRES/RO
//
```

File 3: [EDIT.MAC]

```
        .TITLE      EDIT Editor Front End Utility
        .IDENT      /V1.00/

        .ENABL      REG,LC
        .NLIST      MEB



;
; Programmer:  NWP
;
; Description:
;
; EDIT is a general purpose front end utility to use with any
; installed MCR editing task.  EDIT is installed as ...EDI
; and is invoked either by the MCR command "EDI " or the DCL
; commands "E" or "E/EDI".  Note, if you use EDIT-11 which is
; installed as "...EDI" you might want to install EDIT under
; another task name.  EDIT provides the following features:
;
; 1) Defaults file specification to the last one edited if it
;    is not specified on the command line.
;
; 2) Tacks on a user specified default file extension if none
;    is entered on the command line and if there is no
;    period in the file specification.  ".CMD" is used by
;    default if there is no definition.
;
; 3) Starts up an editor of your choice.  EDT is the default.
;
; EDIT uses a memory file (EDITOR.DEF;1) to save the current
; file specification and user specified defaults.  The format
; of this file is as follows:
;
; Line          Meaning             Default       Size
; ----          -------             -------       ----
;
;   1     Last file spec. edited     Last    Var. max. 80 Char.
;   2     Default file extension     .CMD    Var. max. 4 Char.
;   3     Editor task to use         EDT     Fixed at 3 Char.
;
; If a memory file is not present, EDIT will prompt for a file
; specification and create a new memory file with lines 2 and
; 3 defaulted to ".CMD" and "EDT".  If you edit the memory
; file, make sure that you specify a version number of "1",
; because EDIT will not take the latest version number, that
; is, ";0".  You can enter any valid MCR editor that you want
; on line three, for example EDT, KED, or K52.
;

        .PAGE
        .SBTTL    Local symbol definitions

        .MCALL    DIR$,EXIT$S,GMCR$,MRKT$C,QIOW$C,QIOW$S,RPOI$C,STSE$C
        .MCALL    FSRSZ$,FDBDF$,FINIT$,NMBLK$,FDAT$A,FDRC$A,FDOP$A,FDBF$A
        .MCALL    OPEN$R,OPEN$W,CLOSE$,GET$,PUT$
```

```
;
;       Define all local symbols
;

CR      =       15      ; An ASCII carriage return character
ESC     =       33      ; An ASCII escape character
TICKS   =       1       ; Mark time, time units (ticks)
TIOLUN  =       1       ; User's terminal LUN
FCSLUN  =       2       ; FCS disk LUN
TTEFN   =       1       ; Terminal local event flag
FCSEFN  =       2       ; FCS local event flag
MKTEFN  =       3       ; Mark time local event flag
RECORD  =       80.     ; FCS record buffer size


        .PAGE
        .SBTTL  Define impure data and FCS data structures
        .PSECT  IMPURE,RW,D,LCL,REL,CON

;
;       Define all impure data.
;

        .NLIST  BEX
EDITOR: .ASCII  /EDT/           ; Default MCR/DCL editor to use

PROMPT: .ASCII  /File? /        ; MCR/DCL file prompt when no
                                ;   memory file
PRMSIZ= .-PROMPT

ADDMSG: .ASCII  /EDIT -- Adding default file extension /
ADDSIZ= .-ADDMSG

DEFBUF: .ASCII  /.CMD/          ; Default file extension MUST
                                ;  follow ADDMSG.
EXTSIZ= .-DEFBUF

EDTMSG: .ASCII  /EDIT -- Editing file /
EDTSIZ= .-EDTMSG
        .LIST   BEX

;
;       Define all FCS data structures.
;

RECBUF: .BLKB   RECORD          ; FCS buffer, RECBUF MUST
                                ;   follow EDTMSG !
        .EVEN

SAVFIL: NMBLK$  EDITOR,DEF,1,SY,0 ; Saved memory file specification

        FSRSZ$  1,,IMPURE       ; FCS file store region

;
;       Define the input FDB.
;
```

```
INPFDB:  FDBDF$                          ; Define the input file desc. block.
         FDRC$A   ,RECBUF,RECORD         ; Define the file open sect. of the FDB.
         FDOP$A   FCSLUN,,SAVFIL         ; Init. file open sect. of the FDB.
         FDBF$A   FCSEFN                 ; Init. block buffer sect. of the FDB.

         ;
         ;       Define the output FDB.
         ;

OUTFDB:  FDBDF$                          ; Define the input file desc. block.
         FDAT$A   R.VAR,FD.CR            ; Define the file attributes sect. of FDB.
         FDRC$A   ,RECBUF,RECORD         ; Define the file open sect. of the FDB.
         FDOP$A   FCSLUN,,SAVFIL         ; Init. file open sect. of the FDB.
         FDBF$A   FCSEFN                 ; Init. block buffer sect. of the FDB.

         ;
         ;       Define general impure storage.
         ;

GMCR:    GMCR$                           ; Get MCR command line DPB.
IO.SB:   .BLKW    2                      ; Terminal I/O status block
FILPTR:  .WORD    0                      ; File spec. offset into command buffer
RECSIZ:  .WORD    0                      ; MCR/DCL command line buffer size
DEFSIZ:  .WORD    EXTSIZ                 ; Size of default extension


         .PAGE
         .SBTTL   Start program logic
         .PSECT   EDIT,RO,I,LCL,REL,CON

         ;
         ;       Start the program logic.
         ;

EDIT::
         FINIT$                          ; Initialize the FCS file storage
                                         ;   region.
         QIOW$C   IO.ATT,TI0LUN,TTEFN,,,,,EDIT,ERROR
         DIR$     #GMCR,ERROR            ; Get the MCR command line.
         MOV      #GMCR+G.MCRB,R0        ; Get the buffer address in R0.
         JSR      PC,GETMEM              ; Get last file spec., default ext.
                                         ;   and editor.
         MOV      EDITOR,(R0)+           ; Get the editor to use.  The default
                                         ;   is EDT.
         MOVB     EDITOR2,(R0)+          ; Move the last character.


         .PAGE
         .SBTTL   Parse the MCR/DCL command line

         ;
         ;       Find either a delimiter to a file spec. or a
         ;         line terminator.
         ;       Note, you want to skip over command switches.
         ;
```

```
PARSE:  CMPB    #' ,(R0)        ; Have you hit the first delimiter?
        BEQ     GETNAM          ; If you have a space, then get the
                                ;   file specification.

        CMPB    #CR,(R0)        ; Did you hit a line terminator?
        BEQ     NOSPEC          ; If so, then get a file somewhere.

        CMPB    #ESC,(R0)       ; Did you hit a line terminator?
        BEQ     NOSPEC          ; If so, then get a file somewhere.

        INC     R0              ; Bump the parser pointer.
        BR      PARSE           ; You have to hit something.

        .PAGE
        .SBTTL  Strip off the file specification and save it

        ;
        ;       You have a file specification, so get it.
        ;       Note, you want to stop at either a switch "/",
        ;       a file transfer sign "=", or a line terminator.
        ;

GETNAM: INC     R0              ; Skip over the space.
        CLR     R1              ; Clear the file extension flag.
        CLR     R2              ; Clear the file version flag.
        MOV     R0,FILPTR       ; Save the starting file position.

10$:    CMPB    #'.,(R0)        ; Do you have a file extension?
        BNE     20$             ; If none, then continue parsing.

        INC     R1              ; Set file extension flag to true.
        BR      60$             ; Skip over useless actions.

20$:    CMPB    #';,(R0)        ; Do you have a file version?
        BNE     30$             ; If none, then continue parsing.

        INC     R2              ; Set the file version flag to true.
        BR      60$             ; Skip over useless actions.

30$:    CMPB    #'=,(R0)        ; Did you hit an equal sign?
        BNE     40$             ; If so, then get out.

        JMP     SAVE            ; Take the file spec. as is.

40$:    CMPB    #'/,(R0)        ; Did you hit a switch?
        BNE     50$             ; If so, then get out.

        JMP     SAVE            ; Take the file spec. as is.

50$:    CMPB    #CR,(R0)        ; Did you hit a line terminator?
        BEQ     DONE            ; If so, then get out.

        CMPB    #ESC,(R0)       ; Did you hit a line terminator?
        BEQ     DONE            ; If so, then get out.

60$:    INC     R0              ; Bump the parser pointer.
        BR      10$             ; You have to hit a terminator some
                                ;   time.
```

```
;
;       See if you need to add on a default extension.
;

DONE:   TST     R1              ; Do you have a file extension?
        BEQ     10$             ; Branch if no extension present.

        JMP     SAVE            ; Take the file spec. as is.

10$:    TST     R2              ; Do you have a version specified?
        BEQ     20$             ; Branch if no version number present.

        JMP     SAVE            ; Take the file spec. as is.

        .PAGE

;
;       Tack on a default file extension.
;

20$:    MOV     #DEFBUF,R1      ; Get the addr. of the default extension
        MOV     DEFSIZ,R2       ; Get the default extension size.

30$:    MOVB    (R1)+,(R0)+     ; Move it to the MCR/DCL command line.
        SOB     R2,30$          ; Go back and do the next character.

        MOV     #ADDSIZ,R1      ; Get the message size in bytes.
        ADD     DEFSIZ,R1       ; Add the default file name size.
        QIOW$S  #IO.WVB,#TI0LUN,#TTEFN,,,,<#ADDMSG,R1,#40>,ERROR
        MRKT$C  MKTEFN,5,TICKS,,EDIT,ERROR ; Wait a little.
        STSE$C  MKTEFN,EDIT,ERROR
        MOVB    #CR,(R0)        ; Make sure you have a terminator.
        JMP     SAVE            ; Save the file specification.


        .PAGE
        .SBTTL  No file specification, try memory file

;
;       You have no file specification, so try to get one
;                from the saved memory file.
;

NOSPEC: MOVB    #' ,(R0)+       ; Sub a space for a line terminator.
        MOV     R0,FILPTR       ; Save the starting file position.
        TST     RECSIZ          ; See if you got back a file spec.
        BNE     10$             ; Tack on the file spec. if you have one.
        JMP     GETFIL          ; Go get the file spec. No memory file.

10$:    MOV     RECSIZ,R1       ; Get the record size in R1.
        ADD     #EDTSIZ,R1      ; Get the total message size.
        QIOW$S  #IO.WVB,#TI0LUN,#TTEFN,,,,<#EDTMSG,R1,#40>,ERROR
        MRKT$C  MKTEFN,5,TICKS,,EDIT,ERROR ; Wait a little.
        STSE$C  MKTEFN,EDIT,ERROR
        MOV     #RECBUF,R1      ; Get the memory file starting address.
        MOV     RECSIZ,R2       ; Get the memory file size in bytes.
```

```
20$:    MOVB    (R1)+,(R0)+         ; Move each character to the
                                    ;   command line.
        SOB     R2,20$              ; Go back up and do the next character.

        MOVB    #CR,(R0)            ; Make sure you have a terminator.
        JMP     SAVE                ; Save the file specification.

        ;
        ;       You have no memory file spec. saved, so prompt for it.
        ;

GETFIL: QIOW$S  #IO.RPR,#TI0LUN,#TTEFN,,#IO.SB, -
                        ,<R0,#75.,,#PROMPT,#PRMSIZ,#44>,ERROR
        CMPB    #IE.EOF,IO.SB       ; Did the user type a CTRL/Z?
        BEQ     EXIT                ; If there is a CTRL/Z, dump out nicely.

        ADD     IO.SB+2,R0          ; Point to the end of the command line.
        MOVB    #CR,(R0)            ; Tack on a command line terminator.
        MOV     #GMCR+G.MCRB+3,R0   ; Point to the start of the
                                    ;   command line.
        JMP     PARSE               ; Go back up and parse the file
                                    ;   specification.

        ;
        ;       Write out a new saved memory file specification.
        ;

SAVE:   SUB     #GMCR+G.MCRB+4,R0 ; Calculate the file spec. size.
        MOV     R0,RECSIZ           ; Pass the file specification size.
        JSR     PC,PUTMEM           ; Write out the new memory file spec.
        QIOW$C  IO.DET,TI0LUN,TTEFN,,,,,EDIT,ERROR
        RPOI$C  MCR...,,,,,,,,GMCR+G.MCRB,80.,RP.OEX,,,,,EDIT,ERROR

EXIT:   EXIT$S                      ; You should never get here, but just
                                    ;   in case!

        ;
        ;       Error routine to dump the DSW, IO.SB and PC on error.
        ;

ERROR:  MOV     $DSW,R0             ; Dump the DSW.
        MOV     IO.SB+2,R1          ; Dump the I/O status block.
        MOV     (SP),R2             ; Dump the erroring PC.
        TRAP                        ; Crash. . .


        .PAGE
        .SBTTL  Get memory file specification subroutine

        ;
        ;       Subroutine to get previous memory file specification
        ;                       All registers are saved.
        ;
        ;       Returned:
        ;
        ;         RECBUF --> Address of the FCS record buffer
        ;         DEFBUF --> Address of the default extension buffer
        ;         RECSIZ --> Size of the file specification
        ;         DEFSIZ --> Size of the default file extension
        ;         EDITOR --> Address of editor name to start up
        ;
```

```
GETMEM:  MOV      R0,-(SP)           ; Save R0 for use by FCS.
         CLR      RECSIZ             ; Initially no file size at all.
         OPEN$R   #INPFDB            ; Open the file.
         BCS      20$                ; If error, then no memory file.

         GET$                        ; Get the file specification.
         BCS      20$                ; If error, then EOF!

         MOV      F.NRBD(R0),RECSIZ  ; Return the size of the file spec.
         GET$     ,#DEFBUF,DEFSIZ    ; Get the default extension, if any.
         BCS      20$                ; If error, then EOF!

         TST      F.NRBD(R0)         ; Make sure you got something.
         BEQ      10$                ; Branch if blank line.

         MOV      F.NRBD(R0),DEFSIZ  ; Return the size of the default ext.

10$:     GET$     ,#EDITOR,#3        ; Get the desired editor, if any.

20$:     CLOSE$                      ; Close the file.
         MOV      (SP)+,R0           ; Restore R0; you are done with it.
         RTS      PC                 ; Return with or without a file spec.


         .PAGE
         .SBTTL   Write memory file specification subroutine

         ;
         ;        Subroutine to write current file specification to
         ;          save file
         ;                           All registers are saved.
         ;
         ;        Known:
         ;
         ;                  GMCR   --> Address of command line buffer
         ;                  FILPTR --> Offset into command buffer for
         ;                               file specification
         ;                  DEFBUF --> Address of default extension buffer
         ;                  RECSIZ --> Size of the file specification
         ;                  DEFSIZ --> Size of the default file extension
         ;                  EDITOR --> Address of editor name to start up
         ;

PUTMEM:  MOV      R0,-(SP)           ; Save R0 for use by FCS.
         OPEN$W   #OUTFDB            ; Open the file.
         PUT$     ,FILPTR,RECSIZ     ; Write out the new file specification.
         PUT$     ,#DEFBUF,DEFSIZ    ; Write out the default extension.
         PUT$     ,#EDITOR,#3        ; Write out the personal editor.
         CLOSE$                      ; Close the file.
         MOV      (SP)+,R0           ; Restore R0; you are done with it.
         RTS      PC                 ; Return with or without a file spec.

         .END     EDIT
```

Appendix G
DEC Multinational Character Set

The DEC Multinational Character set is an 8-bit character set with 256 characters. Each character is assigned a decimal equivalent number. These numbers range from 0 to 255. The first 128 characters in the set correspond to the ASCII character set.

Graphics shown in parentheses are ASCII control characters. These are produced on most terminals by pressing the key indicated while holding down the CONTROL key. Characters with numbers greater than 127 can only be entered on VT100 and VT52 type terminals from a screen mode. For terminals with LK201 keyboards, you can use the compose sequences to have the graphic symbol for characters 128 through 255 appear on the screen. For other VT100-type and VT52 terminals, you must use the keypad SPECINS function or the nokeypad ASC command to enter these characters in your text; EDT then displays the EDT symbol that corresponds to the character, not the character graphic.

Graphic	EDT Symbol	Decimal Value	Abbrev.	Description
(@)	^@	0	NUL	null character
(A)	^A	1	SOH	start of heading
(B)	^B	2	STX	start of text
(C)	^C	3	ETX	end of text
(D)	^D	4	EOT	end of transmission
(E)	^E	5	ENQ	enquiry
(F)	^F	6	ACK	acknowledge
(G)	^G	7	BEL	bell
(H)	^H	8	BS	backspace
(I)		9	HT	horizontal tabulation
(J)	<LF>	10	LF	line feed
(K)	<VT>	11	VT	vertical tabulation
(L)	<FF>	12	FF	form feed
(M)	<CR>	13	CR	carriage return
(N)	^N	14	SO	shift out
(O)	^O	15	SI	shift in
(P)	^P	16	DLE	data link escape
(Q)	^Q	17	DC1	device control 1
(R)	^R	18	DC2	device control 2
(S)	^S	19	DC3	device control 3
(T)	^T	20	DC4	device control 4

Graphic	EDT Symbol	Decimal Value	Abbrev.	Description
(U)	^U	21	NAK	negative acknowledge
(V)	^V	22	SYN	synchronous idle
(W)	^W	23	ETB	end of transmission block
(X)	^X	24	CAN	cancel
(Y)	^Y	25	EM	end of medium
(Z)	^Z	26	SUB	substitute
([)	<ESC>	27	ESC	escape
(\)	^\	28	FS	file separator
(])	^]	29	GS	group separator
(^)	^^	30	RS	record separator
(_)	^_	31	US	unit separator
		32	SP	space
!	!	33	!	exclamation point
"	"	34	"	quotation marks (double quote)
#	#	35	#	number sign
$	$	36	$	dollar sign
%	%	37	%	percent sign
&	&	38	&	ampersand
'	'	39	'	apostrophe (single quote)
(	(	40	(	opening parenthesis
)	)	41	)	closing parenthesis
*	*	42	*	asterisk
+	+	43	+	plus
,	,	44	,	comma
-	-	45	-	hyphen or minus
.	.	46	.	period or decimal point
/	/	47	/	slash
0	0	48	0	zero
1	1	49	1	one
2	2	50	2	two
3	3	51	3	three
4	4	52	4	four
5	5	53	5	five
6	6	54	6	six
7	7	55	7	seven
8	8	56	8	eight
9	9	57	9	nine
:	:	58	:	colon
;	;	59	;	semicolon
<	<	60	<	less than
=	=	61	=	equals
>	>	62	>	greater than
?	?	63	?	question mark
@	@	64	@	commercial at
A	A	65	A	uppercase A
B	B	66	B	uppercase B
C	C	67	C	uppercase C
D	D	68	D	uppercase D
E	E	69	E	uppercase E
F	F	70	F	uppercase F
G	G	71	G	uppercase G
H	H	72	H	uppercase H

Graphic	EDT Symbol	Decimal Value	Abbrev.	Description
I	I	73	I	uppercase I
J	J	74	J	uppercase J
K	K	75	K	uppercase K
L	L	76	L	uppercase L
M	M	77	M	uppercase M
N	N	78	N	uppercase N
O	O	79	O	uppercase O
P	P	80	P	uppercase P
Q	Q	81	Q	uppercase Q
R	R	82	R	uppercase R
S	S	83	S	uppercase S
T	T	84	T	uppercase T
U	U	85	U	uppercase U
V	V	86	V	uppercase V
W	W	87	W	uppercase W
X	X	88	X	uppercase X
Y	Y	89	Y	uppercase Y
Z	Z	90	Z	uppercase Z
[	[	91	[	opening bracket
\	\	92	\	back slash
]	]	93	]	closing bracket
^	^	94	^	circumflex
_	_	95	_	underline (underscore)
`	`	96	`	grave accent
a	a	97	a	lowercase a
b	b	98	b	lowercase b
c	c	99	c	lowercase c
d	d	100	d	lowercase d
e	e	101	e	lowercase e
f	f	102	f	lowercase f
g	g	103	g	lowercase g
h	h	104	h	lowercase h
i	i	105	i	lowercase i
j	j	106	j	lowercase j
k	k	107	k	lowercase k
l	l	108	l	lowercase l
m	m	109	m	lowercase m
n	n	110	n	lowercase n
o	o	111	o	lowercase o
p	p	112	p	lowercase p
q	q	113	q	lowercase q
r	r	114	r	lowercase r
s	s	115	s	lowercase s
t	t	116	t	lowercase t
u	u	117	u	lowercase u
v	v	118	v	lowercase v
w	w	119	w	lowercase w
x	x	120	x	lowercase x
y	y	121	y	lowercase y
z	z	122	z	lowercase z
{	{	123	{	opening brace
|	|	124	|	vertical line
}	}	125	}	closing brace
~	~	126	~	tilde

Graphic	EDT Symbol	Decimal Value	Abbrev.	Description
DEL	<DEL>	127	DEL	delete, rubout
	<X80>	128	---	[reserved]
	<X81>	129	---	[reserved]
	<X82>	130	---	[reserved]
	<X83>	131	---	[reserved]
	<IND>	132	IND	index
	<NEL>	133	NEL	next line
	<SSA>	134	SSA	start of selected area
	<ESA>	135	ESA	end of selected area
	<HTS>	136	HTS	horizontal tab set
	<HTJ>	137	HTJ	horizontal tab set with justification
	<VTS>	138	VTS	vertical tab set
	<PLD>	139	PLD	partial line down
	<PLU>	140	PLU	partial line up
	<RI>	141	RI	reverse index
	<SS2>	142	SS2	single shift 2
	<SS3>	143	SS3	single shift 3
	<DCS>	144	DCS	device control string
	<PU1>	145	PU1	private use 1
	<PU2>	146	PU2	private use 2
	<STS>	147	STS	set transmit state
	<CCH>	148	CCH	cancel character
	<MW>	149	MW	message waiting
	<SPA>	150	SPA	start of protected area
	<EPA>	151	EPA	end of protected area
	<X98>	152	---	[reserved]
	<X99>	153	---	[reserved]
	<X9A>	154	---	[reserved]
	<CSI>	155	CSI	control sequence introducer
	<ST>	156	ST	string terminator
	<OSC>	157	OSC	operating system command
	<PM>	158	PM	privacy message
	<APC>	159	APC	application program command
	<XA0>	160	---	[reserved]
¡	<!!>	161	¡	inverted exclamation mark
¢	<C/>	162	¢	cent sign
£	<L->	163	£	pound sign
	<XA4>	164	---	[reserved]
¥	<Y->	165	¥	yen sign
	<XA6>	166	---	[reserved]
§	<S0>	167	§	section sign
¤	<X0>	168	¤	general currency sign
©	<C0>	169	©	copyright sign
ª	<a_>	170	ª	feminine ordinal indicator
«	<<<>	171	«	angle quotation mark left
	<XAC>	172	---	[reserved]
	<XAD>	173	---	[reserved]
	<XAE>	174	---	[reserved]
	<XAF>	175	---	[reserved]
°	<0^>	176	°	degree sign
±	<+->	177	±	plus/minus sign
²	<2^>	178	²	superscript 2

Graphic	EDT Symbol	Decimal Value	Abbrev.	Description
³	<3^>	179	³	superscript 3
	<XB4>	180	---	[reserved]
µ	</U>	181	µ	micro sign
¶	<P!>	182	¶	paragraph sign, pilcrow
·	<.^>	183	·	middle dot
	<XB8>	184	---	[reserved]
¹	<1^>	185	¹	superscript 1
º	<o_>	186	º	masculine ordinal indicator
»	<>>>	187	»	angle quotation mark right
¼	<14>	188	¼	fraction one quarter
½	<12>	189	½	fraction one half
	<XBE>	190	---	[reserved]
¿	<??>	191	¿	inverted question mark
À	<A`>	192	À	uppercase A with grave accent
Á	<A´>	193	Á	uppercase A with acute accent
Â	<A^>	194	Â	uppercase A with circumflex
Ã	<A~>	195	Ã	uppercase A with tilde
Ä	<A">	196	Ä	uppercase A with umlaut,(diaeresis)
Å	<A*>	197	Å	uppercase A with ring
Æ	<AE>	198	Æ	uppercase AE diphthong
Ç	<C,>	199	Ç	uppercase C with cedilla
È	<E`>	200	È	uppercase E with grave accent
É	<E´>	201	É	uppercase E with acute accent
Ê	<E^>	202	Ê	uppercase E with circumflex
Ë	<E">	203	Ë	uppercase E with umlaut, (diaeresis)
Ì	<I`>	204	Ì	uppercase I with grave accent
Í	<I´>	205	Í	uppercase I with acute accent
Î	<I^>	206	Î	uppercase I with circumflex
Ï	<I">	207	Ï	uppercase I with umlaut, (diaeresis)
	<XD0>	208	---	[reserved]
Ñ	<N~>	209	Ñ	uppercase N with tilde
Ò	<O`>	210	Ò	uppercase O with grave accent
Ó	<O´>	211	Ó	uppercase O with acute accent
Ô	<O^>	212	Ô	uppercase O with circumflex
Õ	<O~>	213	Õ	uppercase O with tilde
Ö	<O">	214	Ö	uppercase O with umlaut, (diaeresis)
Œ	<OE>	215	Œ	uppercase OE ligature
Ø	<O/>	216	Ø	uppercase O with slash
Ù	<U`>	217	Ù	uppercase U with grave accent
Ú	<U´>	218	Ú	uppercase U with acute accent
Û	<U^>	219	Û	uppercase U with circumflex
Ü	<U">	220	Ü	uppercase U with umlaut, (diaeresis)
Ÿ	<Y">	221	Ÿ	uppercase Y with umlaut, (diaeresis)
	<XDE>	222	---	[reserved]
ß	<ss>	223	ß	German lowercase sharp s
à	<a`>	224	à	lowercase a with grave accent
á	<a´>	225	á	lowercase a with acute accent
â	<a^>	226	â	lowercase a with circumflex
ã	<a~>	227	ã	lowercase a with tilde
ä	<a">	228	ä	lowercase a with umlaut, (diaeresis)
å	<a*>	229	å	lowercase a with ring
æ	<ae>	230	æ	lowercase ae diphthong

Graphic	EDT Symbol	Decimal Value	Abbrev.	Description
ç	<c,>	231	ç	lowercase c with cedilla
è	<e`>	232	è	lowercase e with grave accent
é	<e´>	233	é	lowercase e with acute accent
ê	<e^>	234	ê	lowercase e with circumflex
ë	<e">	235	ë	lowercase e with umlaut, (diaeresis)
ì	<i`>	236	ì	lowercase i with grave accent
í	<i´>	237	í	lowercase i with acute accent
î	<i^>	238	î	lowercase i with circumflex
ï	<i">	239	ï	lowercase i with umlaut, (diaeresis)
	<XF0>	240	---	[reserved]
ñ	<n~>	241	ñ	lowercase n with tilde
ò	<o`>	242	ò	lowercase o with grave accent
ó	<o´>	243	ó	lowercase o with acute accent
ô	<o^>	244	ô	lowercase o with circumflex
õ	<o~>	245	õ	lowercase o with tilde
ö	<o">	246	ö	lowercase o with umlaut, (diaeresis)
œ	<oe>	247	œ	lowercase oe ligature
ø	<ø>	248	ø	lowercase o with slash
ù	<u`>	249	ù	lowercase u with grave accent
ú	<u´>	250	ú	lowercase u with acute accent
û	<u^>	251	û	lowercase u with circumflex
ü	<u">	252	ü	lowercase u with umlaut, (diaeresis)
ÿ	<y">	253	ÿ	lowercase y with umlaut, (diaeresis)
	<XFE>	254	---	[reserved]
	<XFF>	255	---	[reserved]

Index

Entries in this index have been marked to reflect the fact that EDT has three different editing modes and works with different operating systems. The indicators for the editing modes are: (K) for keypad, (L) for line, and (N) for nokeypad. The operating system indicators are (RSTS/E), (RSX-11M/M+), and (VAX/VMS). Entries that have the same name in more than one editing mode or operating system are listed separately. For example, FILL (K), FILL (L), and FILL (N) are three separate entries indexing the FILL command, which exists in all three editing modes.

☞symbol for system prompt, 2-5
〈X〉key (LK201 keyboard), 2-14, 7-24, 7-29, 8-73
  See also DELETE key
! (exclamation point)
  indicating comments in startup command files, 7-2
  sentence boundary, 6-8
" (double quotation mark)
  delimiter character, 2-11, 4-2, 4-6, 5-4, 5-6, 5-30, 6-1, 6-6, 7-14, 7-17 to 7-19, 7-26, 7-27
  in key definitions, 7-19, 7-31
  string delimiter, 4-6, 5-30
  with prompt text in key definitions, 7-17 to 7-19
  with SET ENTITY (L), 6-6
  with string specifier (L), 6-1
n (number sign digit) range specifier symbol (L), 4-5
% (percent sign), 4-19
  with range specifier (L), 4-9, 4-16, 8-132
( ) [parentheses]
  in key definitions, 7-16 to 7-17
  with nokeypad commands, 5-42 to 5-45
* (asterisk)
  in key definitions, 7-17
  in SHOW BUFFER (L) display, 6-12
  line mode prompt, 2-5, 2-16, 3-9, 3-59, 4-2, 5-14
  wildcard character with HELP (L), 4-12
+ (plus sign)
  range specifier symbol (L), 4-5
  sign specifier (N), 5-4
+ | – (sign) specifier (N), 5-2 to 5-6
  with APPEND (N), 5-35 to 5-36, 8-8
  with CHGC (change case) (N), 5-27, 8-21
  with CHGL (change case lower) (N), 5-27, 8-22
  with CHGU (change case upper) (N), 5-27, 8-23
  with CUT (N), 5-31, 8-61
  with D (delete) (N), 5-20 to 5-22, 8-62
  with FILL (N), 5-40 to 5-41, 8-104
  with "move" (N), 5-18 to 5-20, 8-129
  with R (replace) (N), 5-24, 8-151
  with S (substitute) (N), 5-24 to 5-26, 8-165
  with SN (substitute next) (N), 5-26 to 5-27, 8-245
  with SSEL (search and select) (N), 5-29, 5-30, 8-248
  with TADJ (tab adjust) (N), 8-267
' (single quotation mark)
  delimiter character, 2-11, 4-2, 4-6, 5-4, 5-6, 5-30, 6-1, 6-6, 7-14, 7-17 to 7-19, 7-26, 7-27
  in key definitions, 7-19, 7-31
  string delimiter, 4-6, 5-30
  with prompt text in key definitions, 7-17 to 7-19
  with SET ENTITY (L), 6-6
  with string specifier (L), 6-1
, (comma) range specifier symbol (L), 4-5
– (minus sign)
  range specifier symbol (L), 4-5
  sign specifier (N), 5-4
  with string range specifier (L), 4-6
  with TAB ADJUST (L), 7-74 to 7-75

. (period)
signal to process key definitions, 7-16, 7-27
sentence boundary, 6-8
range specifier (L), 4-5 to 4-6
with FIND (L) and buffer specifier (L), 4-29
with <null> (L), 4-30
with TYPE (L), 4-29
/ (slash), 4-47
qualifier indicator (L), 4-3, 4-46
SUBSTITUTE (L) delimiter, 4-19
: (colon)
range specifier symbol (L), 4-5
with increment and initial specifiers, 7-82
with qualifier (RSX-11M/M+), F-1
with qualifier (VAX/VMS), D-2
; (semicolon)
with INSERT (L) line-to-be-inserted, 4-37
with multicommand line (L), 4-45, 7-52, 7-54
with REPLACE (L) line-to-be-inserted, 4-38
< > (angle brackets) – control character symbol, 3-55
= (equal sign)
buffer signal (L), 4-4, 4-28, 8-16
buffer signal (N), 5-3, 5-31, 5-32, 5-35, 8-16
current buffer indicator with SHOW BUFFER (L), 2-8, 6-12
with qualifier (RSTS/E), E-2
with qualifier (RSX-11M/M+), F-1
with qualifier (VAX/VMS), D-2
? (question mark)
prompt for /QUERY qualifier (L), 2-5, 4-47 to 4-48
prompt in key definitions, 7-17 to 7-19, 7-31
sentence boundary, 6-8
[ ] (square brackets)
cursor indicator, hardcopy change mode, 7-76
enclosing command options, 4-3, 5-3
^ (circumflex) control character symbol, 3-54
^ (circumflex) (N), 2-10, 5-11, 5-18, 5-41 to 5-43, 7-76, 8-27
with character specifier, 5-42 to 5-43
with count specifier, 5-42 to 5-43
^@ (null character) – CTRL/@, 5-41
string delimiter in key definitions, 7-18
^Z (CTRL/Z) (L), 4-13
^Z (CTRL/Z) (N), 5-18
_ (underscore), 4-19
in buffer names, 4-4

A

A (all), /QUERY qualifier response (L), 4-47 to 4-48
Abbreviation
line mode commands, 4-2 to 4-3
line mode qualifiers, 4-3, 4-46
Active select range, 3-38, 3-46, 5-29, 5-30
Adding lines in keypad mode, 3-30 to 3-31
ADV (advance) (N), 5-4, 5-6, 5-11, 5-37 to 5-38, 8-4
ADVANCE (K), 3-14, 3-32 to 3-34, 7-21, 8-5
Advanced video option (AVO), C-1
All – A, /QUERY qualifier response (L), 4-47 to 4-48
ALL range specifier symbol (L), 4-5, 4-6
Alternate function, 3-3, 3-6
AND range specifier symbol (L), 4-5
Angle brackets (< >) – control character symbol, 3-55
APPEND (K), 3-36, 3-42 to 3-46, 7-21, 8-6
APPEND (N), 5-11, 5-31, 5-35 to 5-36, 8-8
with buffer specifier, 5-3, 5-35 to 5-36
with count specifier, 5-35 to 5-36
with entity specifier, 5-35 to 5-36
with PASTE buffer, 5-3
with sign (+ | –) specifier, 5-35 to 5-36
Arrow keys, 2-14, 3-1, 3-3, 5-1, 5-3
See also Down Arrow, Left Arrow, Right Arrow, Up Arrow
ASC (ASCII) (N), 2-10, 5-11, 5-41 to 5-42, 7-2, 8-9
with count specifier, 5-42
with number specifier, 5-41 to 5-42
with SET ENTITY (L), 6-6
ASCII control characters, 5-41, 8-27
Asterisk (*)
in key definitions, 7-17
in SHOW BUFFER (L) display, 6-12
line mode prompt, 2-5, 2-16, 3-9, 3-59, 4-2, 5-14
wildcard with HELP (L), 4-12
Asynchronous terminals, C-1
Attributes of files, D-7, E-2, F-7
Autorepeat suppression in terminals, D-6, E-9, F-9
AVO (advanced video option), C-1

B

B entity prefix (N), 5-4
BACK (N), 5-4, 5-6, 5-11, 5-18, 5-37 to 5-38, 8-10
Backing up
  keypad mode, 3-14, 3-55 to 3-57
  nokeypad mode, 5-4, 5-18, 5-37 to 5-38
BACKSPACE (K), 2-14, 3-14, 3-15, 7-14 to 7-15, 7-23, 7-25, 7-28, 8-11
BACKSPACE (L), 2-14
BACKSPACE (N), 2-14
BACKSPACE key, 3-2, 3-3, 6-14
BACKUP (K), 3-14, 3-32 to 3-34, 7-21, 8-12
Backward direction, 2-10, 3-14, 5-37 to 5-38
Backward function
  keypad mode, 3-55 to 3-57
.BAK file type, E-2
BEFORE range specifier (L), 4-5, 4-8
BEGIN range specifier (L), 4-5, 4-7
Beginning of line (BL) entity specifier (N), 5-5
Beginning of page (BPAGE) entity specifier (N), 5-6
Beginning of paragraph (BPAR) entity specifier (N), 5-6
Beginning of range (BR) entity specifier (N), 5-6
Beginning of sentence (BSEN) entity specifier (N), 5-6
Beginning of word (BW) entity specifier (N), 5-5
BELL (N), 5-11, 5-40, 8-13
BL (beginning of line) entity specifier (N), 5-5
Block, 7-81
BOTTOM (K), 3-14, 3-15 to 3-16, 7-21, 8-14
Bottom specifier (L)
  with SET CURSOR (L), 8-180
Boundary, 5-6
  page, 6-9
  paragraph, 6-8
  sentence, 6-8
  word, 3-20, 6-7 to 6-8
BPAGE (beginning of page) entity specifier (N), 5-6
BPAR (beginning of paragraph) entity specifier (N), 5-6
BR (beginning of range) entity specifier (N), 5-6
Brackets
  angle (< >) – control character symbol, 3-55
  square ([ ])
    cursor indicator, hardcopy change mode, 7-76
    enclosing command options, 4-3, 5-3
/BRIEF qualifier (L), 4-46 to 4-47, 8-15
  with SUBSTITUTE (L), 4-19, 4-46 to 4-47, 8-255
  with TYPE (L), 4-16, 4-46 to 4-47, 8-277
BSEN (beginning of sentence) entity specifier (N), 5-6
Buffer, 2-3, 2-7, 4-28
  current, 2-8, 3-37, 4-4, 4-28, 4-29, 4-32 to 4-35, 4-37, 4-40, 4-41, 4-43, 4-49, 5-3, 5-24, 5-32, 5-39, 6-12, 7-74
  delete character, 2-8, 3-18 to 3-20, 5-22 to 5-24, 6-12
  delete line, 2-8, 3-24 to 3-29, 5-22 to 5-24, 6-12
    line terminator in, 5-23
  delete word, 2-8, 3-20 to 3-24, 5-22 to 5-24, 6-12
    space in, 5-23
  deleting with CLEAR (L), 4-30 to 4-31, 8-28
  MAIN, 2-7, 2-8, 2-15, 2-16, 3-8, 4-4, 4-10, 4-28, 4-29, 4-31, 4-36, 6-12, 7-81 to 7-83
    deleting contents with CLEAR (L), 2-8, 4-31
  mark for end of – [EOB], 2-9
  moving to another, 4-4
  PASTE, 2-8, 3-37 to 3-46, 3-49 to 3-52, 4-31, 4-36, 5-31, 5-32, 5-35, 6-12
    deleting contents with CLEAR (L), 2-8, 3-38, 4-31, 6-13
    with APPEND (N), 5-3
    with CUT (N), 5-3
    with PASTE (N), 5-3
  search, 2-8, 3-32 to 3-36, 3-49 to 3-52, 4-19, 5-25, 5-26, 5-30, 5-40, 6-12
  substitute, 2-8, 4-19, 5-26, 6-12
  with EDT macro, 7-42
  with SHOW BUFFER (L), 6-12 to 6-14
BUFFER – buffer signal (L), 4-4, 4-28, 8-16
Buffer name, 2-8, 4-4, 4-28, 4-29
  underscore (_) character, 4-4
  with CLEAR (L), 4-30
Buffer signal
  BUFFER (L), 4-4, 4-28, 8-16
  equal sign (=) (L), 4-4, 4-28, 8-16
  equal sign (=) (N), 5-3, 5-31, 5-32, 5-35, 8-16
Buffer specifier (L), 2-8, 4-4, 4-28, 4-30, 8-16
  with CHANGE (L), 5-13, 8-18
  with CLEAR (L), 4-4, 4-28, 4-30 to 4-31, 8-28
  with DELETE (L), 4-36, 8-75
  with FILL (L), 4-43 to 4-44, 8-102
  with FIND (L), 4-16 to 4-18, 4-30, 8-108
    period (.) range specifier (L), 4-29
  with INCLUDE (L), 4-39, 8-120

Buffer specifier (L), (cont.)
with INSERT (L), 4-37, 8-121
with <null> (L), 4-16 to 4-18, 4-30, 8-132
with PRINT (L), 4-4, 4-28, 4-41, 8-146
with /QUERY qualifier (L), 4-47
with REPLACE (L), 4-35 to 4-36, 4-38, 8-157
with RESEQUENCE (L), 4-41 to 4-43, 8-159
with SUBSTITUTE (L), 4-19 to 4-21, 4-22 to 4-23, 8-255
with SUBSTITUTE NEXT (L), 4-19, 4-21 to 4-23
with TAB ADJUST (L), 7-74 to 7-75, 8-265
with TYPE (L), 4-16 to 4-18, 4-30, 8-277
with WRITE (L), 4-4, 4-28, 4-40, 8-292
Buffer specifier (N), 2-8, 5-2, 5-3, 8-16
with APPEND (N), 5-3, 5-35 to 5-36, 8-8
with CUT (N), 5-3, 5-31, 8-61
with PASTE (N), 5-3, 5-32 to 5-34, 8-143
Buffer-1 specifier (L), 4-4
with COPY (L), 4-33 to 4-35, 8-33
with MOVE (L), 4-31 to 4-33, 8-128
Buffer-2 specifier (L), 4-4
with COPY (L), 4-33 to 4-35, 8-33
with MOVE (L), 4-31 to 4-33, 8-128
BW (beginning of word) entity specifier (N), 5-5

C

C (character) entity specifier (N), 5-5
C* – hardcopy change mode prompt, 7-76
Callable EDT, D-9 to D-17
Calling up EDT, 2-5, 2-15, 3-6, 5-12, D-1, E-1, F-1
Cancelling a search, 3-34
Cancelling a select range, 3-37, 5-30
Cancelling the COMMAND (K) line, 3-58
Carriage return, 3-51
See also line terminator
Case of letters
changing, 3-46 to 3-48, 5-27 to 5-28
in search strings, 2-12, 6-10 to 6-11
CCL EDT command qualifiers (RSTS/E), E-6 to E-9
CCL EDT command specifiers (RSTS/E), E-6 to E-9
CCL monitor with RSTS/E, E-1
/CCL qualifier (RSTS/E), E-8
/CHAIN qualifier (RSTS/E), E-8
Chainable EDT (RSTS/E), E-10 to E-13
CHANGE (L), 3-6, 3-7, 3-59, 4-2, 4-45, 5-12, 5-13, 7-49, 7-50, 7-52, 7-76, 8-18
Change mode, 2-1, 5-13
hardcopy
See hardcopy change mode
shifting from line mode, 5-13
Changing case of letters, 3-46 to 3-48, 5-27 to 5-28
Changing direction
keypad mode, 3-14
nokeypad mode, 5-37 to 5-38
CHAR (K), 3-14, 7-21, 8-20
Character
contiguous, 2-11, 3-36, 4-19, 6-7
control, 2-3, 2-9, 2-10, 3-54 to 3-55, 5-42 to 5-43
ASCII, 5-41
with SET ENTITY (L), 6-6
cursor, 2-10, 3-52, 5-6
changing case of, 3-46 to 3-48
decimal value, 2-10, 3-6, G-1 to G-6
delimiter, 2-11
escape (<ESC>) in journal file, 7-58
string delimiter, 5-4
Character editing, 2-10, 5-1
Character entity, 2-9
Character (C) entity specifier (N), 5-5
Character function
See CHAR (K)
Character set – DEC Multinational, 5-41, G-1 to G-6
Character specifier (N), 5-3
with circumflex (^) (N), 5-42 to 5-43, 8-27
Character symbol, 3-54 to 3-55, G-1
CHGC (change case) (N), 5-11, 5-27, 8-21
with count specifier, 5-27
with entity specifier, 5-27
with sign (+ | –) specifier, 5-27
CHGL (change case lower) (N), 5-11, 5-27, 8-22
with count specifier, 5-27
with entity specifier, 5-27
with sign (+ | –) specifier, 5-27
CHGU (change case upper) (N), 5-11, 5-27, 8-23
with count specifier, 5-27
with entity specifier, 5-27
with sign (+ | –) specifier, 5-27
CHNGCASE (K), 3-46 to 3-48, 7-21, 8-24
Circumflex (^) control character symbol, 3-54
Circumflex (^) (N), 2-10, 5-11, 5-18, 5-41 to 5-43, 7-76, 8-27
with character specifier, 5-42 to 5-43
with count specifier, 5-42 to 5-43

CLEAR (L), 2-8, 3-37, 4-2, 4-36, 6-12, 8-28
deleting a buffer, 4-30 to 4-31
with buffer specifier, 4-4, 4-28, 4-30 to 4-31
CLSS (clear search string) (N), 3-35, 5-11, 5-40, 8-29
Colon (:)
range specifier symbol (L), 4-5
with increment and initial specifiers, 7-82
with qualifier (RSX-11M/M+), F-1
with qualifier (VAX/VMS), D-2
Comma (,) range specifier symbol (L), 4-5
COMMAND (K), 3-9, 3-58 to 3-59, 7-21, 7-51 to 7-52, 8-31
cancelling the line mode command, 3-58
editing the line mode command, 3-58
prompt – Command:, 2-5, 3-9, 3-58, 7-51 to 7-52
Command file
See startup command file
Command line, 4-2, 5-1, 5-11, 5-17
qualifier position (L), 4-3, 4-46
with more than one command (L), 4-45
with more than one command (N), 5-43 to 5-45
Command options – enclosed in square brackets ([ ]), 4-3
/COMMAND qualifier (RSTS/E), E-3
/COMMAND qualifier (RSX-11M/M+), F-2
/COMMAND qualifier (VAX/VMS), D-3
Command word, 4-2
Command:–COMMAND (K) prompt, 2-5, 3-9, 3-58, 7-51 to 7-52
Commands, line mode, 4-2 to 4-3
Commands, nokeypad mode, 5-2 to 5-3
Comments in startup command files, 7-2
Contiguous characters, 2-11, 3-36, 4-19, 6-7
Contiguous lines, 4-2, 4-5
Control (CTRL) key, 2-3, 2-9, 2-10, 3-2, 3-3, 6-14
defining, 7-13 to 7-14
help information (K), 3-10
in HELP file text, 7-88
Control character, 2-3, 2-9, 2-10, 3-54 to 3-55, 5-42 to 5-43
ASCII, 5-41
in key definitions, D-6, E-9, F-8
in startup command files, 7-2
with SET ENTITY (L), 6-6
COPY (L), 4-1, 4-31, 4-33 to 4-35, 8-33
location-1, 4-34 to 4-35
location-2, 4-34 to 4-35
with buffer-1 specifier, 4-33 to 4-35
COPY (L), (cont.)
with buffer-2 specifier, 4-33 to 4-35
with /DUPLICATE qualifier, 4-33 to 4-46
with /QUERY qualifier, 4-33, 4-46 to 4-48
with range-1 specifier, 4-33 to 4-35
with range-2 specifier, 4-33 to 4-35
Copying text
keypad mode, 3-36 to 3-46
line mode, 4-31, 4-33 to 4-35
nokeypad mode, 5-31 to 5-37
Count specifier (N), 5-2, 5-3 to 5-4, 8-35
repeat count limit, 5-3
with APPEND (N), 5-35 to 5-36, 8-8
with ASC (ASCII) (N), 5-42, 8-9
with CHGC (change case) (N), 5-27, 8-21
with CHGL (change case lower) (N), 5-27, 8-22
with CHGU (change case upper) (N), 5-27, 8-23
with circumflex (^) (N), 5-42 to 5-43, 8-27
with CUT (N), 5-31, 8-61
with D (delete) (N), 5-20 to 5-22, 8-62
with FILL (N), 5-40 to 5-41, 8-104
with "move" (N), 5-18 to 5-20, 8-129
with PASTE (N), 5-32 to 5-34, 8-143
with R (replace) (N), 5-24, 8-151
with S (substitute) (N), 5-24 to 5-26, 8-165
with SHL (shift left) (N), 5-39, 8-211
with SHR (shift right) (N), 5-39, 8-244
with SN (substitute next) (N), 5-26 to 5-27, 8-245
with TAB (N), 8-263
with TADJ (tab adjust) (N), 8-267
with TD (tab decrement) (N), 8-270
with TI (tab increment) (N), 8-273
with UNDC (undelete character) (N), 5-22 to 5-24, 8-280
with UNDL (undelete line) (N), 5-22 to 5-24, 8-284
with UNDW (undelete word) (N), 5-22 to 5-24, 8-288
<CR> – line terminator symbol, 7-76
Crash
See system interruption
/CREATE qualifier (RSTS/E), E-4
/CREATE qualifier (RSX-11M/M+), F-2
/CREATE qualifier (VAX/VMS), D-3
Creating a file with EDT, 2-16, 3-7, 3-12, 4-13
Creating a select range, 3-36 to 3-37, 5-29 to 5-30
CTRL (control) key, 2-3, 2-9, 2-10, 3-2, 3-3, 6-14
defining, 7-12 to 7-14
help information (K), 3-10
in HELP file text, 7-88

CTRL/@ – null character (^@), 5-41
  string delimiter in key definitions, 7-18
CTRL/[, D-6
CTRL/A (K), 7-22, 7-62, 7-67 to 7-73, 8-36
CTRL/C, 7-14, D-6, E-9, F-8
CTRL/C (K), 8-38
CTRL/C (L), 4-29, 8-38
CTRL/C (N), 8-38
CTRL/D (K), 7-22, 7-62, 7-67 to 7-73, 8-39
CTRL/E (K), 7-22, 7-62, 7-66 to 7-73, 8-41
CTRL/F (K), 3-48 to 3-49, 7-22, 8-99
CTRL/H (K), 2-14, 6-14, 7-15, 7-22, 7-28, 8-11
CTRL/I (K), 2-14, 6-14, 7-15, 7-22, 7-28, 7-62, 8-261
CTRL/I – horizontal tab, 3-53, 4-44, 7-63
CTRL/J (K), 2-14, 6-14, 7-15, 7-22, 7-28, 8-126
CTRL/K (K), 7-12, 7-20 to 7-26, 7-32 to 7-42, 7-53, 8-43
CTRL/L, 2-9, 3-15, 6-9
CTRL/L (K), 7-22, 8-47
CTRL/M, 3-9, 3-51
CTRL/M (K), 7-15, 7-22, 7-28, 8-49
CTRL/O, D-6, E-9, F-8
CTRL/P, D-6, E-9, F-8
CTRL/Q, 7-14, D-6, E-9, F-8
CTRL/R (K), 3-12, 7-22, 8-51
CTRL/R (L), 8-51
CTRL/S, 7-14, D-6, E-9, F-8
CTRL/T, D-6, E-9
CTRL/T (K), 7-22, 7-62, 7-67 to 7-73, 8-52
CTRL/U (K), 3-18, 3-24 to 3-29, 3-34, 7-22, 8-54
  cancelling COMMAND (K), 3-58
  cancelling CTRL/K (K), 7-26
  cancelling FIND (K), 3-34
CTRL/U (N), 5-11
CTRL/W (K), 3-12, 7-22, 8-55
CTRL/X, 7-14, D-6, E-9, F-8
CTRL/Y, D-6
CTRL/Z (K), 3-9, 3-59 to 3-61, 7-22, 7-50, 7-52, 8-56
CTRL/Z (L), 4-13, 8-57
  with INSERT (L), 4-37, 8-121
  with REPLACE (L), 4-35 to 4-36, 8-157
CTRL/Z (N), 5-17 to 5-18, 5-24, 7-76, 8-57
  with I (insert) (N), 5-17 to 5-18, 8-118
  with R (replace) (N), 5-24, 8-151
CTRL/Z in DEFINE KEY (L) definitions, 7-30
Current buffer, 3-37, 4-4, 4-28, 4-29, 4-32 to 4-35, 4-37, 4-40, 4-41, 4-43, 4-49, 5-3, 5-24, 5-32, 5-39, 6-12, 7-74
  equal sign (=) indicator with SHOW BUFFER (L), 2-8, 6-12
Current cursor position, 5-2, 5-41, 6-7
Current direction, 2-12, 3-14, 5-4, 5-37 to 5-38, 6-7
  with KS (KED substitute) (N)/PASTE (N), 5-36 to 5-37
Current directory, 3-10, 4-10, D-2
  location of journal file, 4-11
Current line, 2-3, 4-4, 4-5, 4-16, 4-30, 4-32, 4-34 to 4-38, 4-49, 5-39
Current screen width, 5-40 to 5-41
Current search string, 3-35
Cursor, 2-3, 2-10, 5-1
  current position, 5-41, 6-7
  inserting text in keypad mode, 3-13
  moving
    keypad mode, 3-1, 3-14 to 3-17
    nokeypad mode, 5-3, 5-18 to 5-20
  position after screen mode search, 2-11, 6-11
  position with KS (KED substitute) (N)/PASTE (N), 5-36 to 5-37
Cursor character, 2-10, 3-52, 5-6
  changing case of, 3-46 to 3-48
Cursor indicator, 2-10
  square brackets ([ ]) – hardcopy change mode, 7-76
CUT (K), 3-36 to 3-46, 7-21, 8-59
  use of PASTE buffer with, 2-8
CUT (N), 5-11, 5-31, 8-61
  with buffer specifier, 5-3, 5-31
  with count specifier, 5-31
  with entity specifier, 5-31
  with PASTE buffer, 2-8, 5-3
  with sign (+ | –) specifier, 5-31

D

D (delete) (N), 5-11, 5-20 to 5-22, 8-62
  with count specifier, 5-20 to 5-22
  with entity specifier, 5-20 to 5-22
  with sign (+ | –) specifier, 5-20 to 5-22
  with string entity specifier, 5-20
Data transmission rate, 5-2, C-2
DATE (N), 5-11, 5-41, 5-43, 8-64
DCL EDIT/EDT (RSTS/E), E-1
  qualifiers, E-3 to E-6
DCL EDIT/EDT (RSX-11M/M+), F-1
  qualifiers, F-1 to F-4
DCL EDIT/EDT (VAX/VMS), D-1
  qualifiers, D-2 to D-5
DEC Multinational Character Set, 3-53 to 3-55, 5-41, G-1 to G-6
DECARM VT100 control sequence, C-1
Decimal fraction line numbers, 2-13, 7-81

Decimal value of character, 2-10, 3-6, 3-53 to 3-55, 5-41, G-1 to G-6
Default, 2-4
  direction, 2-10
  line length, 4-43
  specifier (L), 4-3
DEFINE KEY (L), 4-2, 7-12, 7-20, 7-26 to 7-31, 7-32 to 7-42, 7-53, 8-65
  CTRL/Z in key definition, 7-30
  I (insert) (N) in key definition, 7-30
  keypad key numbers with, 7-27
  with key-name specifier (L), 7-26
  with string specifier, 7-27
DEFINE MACRO (L), 4-2, 7-42 to 7-48, 8-68
  with macro-name specifier, 7-43 to 7-48
Defining keys in keypad mode, 7-11 to 7-42
  CTRL/K (K), 7-20 to 7-26
  DEFINE KEY (L), 7-20, 7-26 to 7-31
  SHOW KEY (L), 6-14 to 6-16
Defining macros, 7-42 to 7-48
Definitions of keypad editing keys, 3-6, 7-11 to 7-42
  nokeypad commands, 7-15 to 7-17
DEFK (define key) (N), 5-11, 7-12, 8-70
DEL C (K), 3-18 to 3-20, 3-25, 7-21, 8-71
DEL EOL (K), 3-18, 3-24 to 3-29, 7-21, 8-72
DEL L (K), 3-18, 3-24 to 3-29, 7-21, 8-77
DEL W (K), 3-18, 3-20 to 3-24, 7-21, 8-79
DELETE (K), 3-18, 3-18 to 3-20, 3-25, 3-34, 3-53, 7-14 to 7-15, 7-23, 7-28, 8-73
  editing COMMAND (K) line, 3-58
  editing GOLD repeat number, 3-56
DELETE (L), 4-1, 4-13 to 4-16, 4-36, 4-36, 8-75
  with buffer specifier, 4-36
  with /QUERY qualifier, 4-36, 4-46 to 4-48
  with range specifier, 4-36
DELETE buffer
  with REPLACE (K), 3-45, 6-13
  with SUBS (K), 3-49, 6-13
Delete character buffer, 2-8, 3-18 to 3-20, 5-22 to 5-24, 6-12
Delete functions (K), 3-18 to 3-30
DELETE key, 2-14, 3-2, 3-3, 6-14
  editing key definitions, 7-26, 7-27
  in HELP file text, 7-88
  keypad mode, 3-18, 3-34
  line mode, 2-14
  nokeypad mode, 5-3, 5-11, 5-17, 5-23
  redefining, 7-12 to 7-13
  screen mode, 2-14
Delete line buffer, 2-8, 3-24 to 3-29, 5-22 to 5-24, 6-12
  line terminator in, 5-23
Delete word buffer, 2-8, 3-20 to 3-24, 5-22 to 5-24, 6-12
  space in, 5-23
Deleting a buffer with CLEAR (L), 2-8, 4-30 to 4-31, 8-28
Deleting characters, 3-18 to 3-20
  See D (delete) (N)
  See DELETE (K) and DEL C (K)
Deleting lines, 3-24 to 3-29
  See D (delete) (N)
  See DELETE (L)
  See DEL L (K), DEL EOL (K), and CTRL/U (K)
Deleting text
  keypad mode, 3-18 to 3-30, 3-36 to 3-46
  line mode, 4-13 to 4-16, 4-31, 4-35 to 4-36
  nokeypad mode, 5-20 to 5-22
Deleting the contents of the search buffer – CLSS (N), 3-35
Deleting the horizontal tab, 3-53
Deleting words, 3-20 to 3-24
  See D (delete) (N)
  See DEL W (K) and LINEFEED (K)
Delimiter, 2-11, 4-2, 4-47
  in substitutions, 2-12, 4-19, 5-4
  quotation marks (' ,"), 2-11, 4-2, 4-6, 5-30
  string, 4-2, 4-6, 5-2, 5-4, 5-30
  SUBSTITUTE (L), 4-47
DESEL (deactivate select) (N), 5-11, 5-29, 5-30, 8-81
Device name, 2-7
Diacritical marks, 3-54
  in search strings, 2-12, 6-10 to 6-11
Direction, 2-3, 2-10, 3-14, 3-37, 5-4, 5-37 to 5-38
  backward, 3-14, 5-37 to 5-38
  changing in keypad mode, 3-14
  changing in nokeypad mode, 5-37 to 5-38
  current, 2-12, 3-14, 5-37 to 5-38, 6-7
    with KS (KED substitute) (N)/PASTE (N), 5-36 to 5-37
  forward, 3-37, 5-37 to 5-38
  search, 3-32 to 3-34
Directory, 2-8, 2-16, 3-10, 4-10 to 4-11, 4-38, 4-40, 4-41
  current, 3-10, 4-10, D-2
  location of journal file, 2-18, 4-11
Directory specification, 2-6, 2-7, 2-16, 4-10 to 4-11
Disk space, 7-81
DLWC (default lowercase) (N), 5-11, 5-27, 5-28, 8-82
DMOV (default move) (N), 5-11, 5-27, 5-28, 8-83

DO (K) (LK201 keyboard), 8-84
Do key (LK201 keyboard), 7-24, 8-84
  See also DO (K)
  See also ENTER (K)
Dot
  See period
Double quotation mark (")
  delimiter character, 2-11, 4-2, 4-6, 5-4, 5-6, 5-30, 6-1, 6-6, 7-14, 7-17 to 7-19, 7-26, 7-27
  in key definitions, 7-31
  with prompt text in key definitions, 7-17 to 7-19
Down Arrow (K), 3-14, 3-15, 7-21, 8-86
Down Arrow (N), 8-86
DUPC (default uppercase) (N), 5-11, 5-27, 5-28, 8-87
/DUPLICATE qualifier (L), 8-88
  repeat count limit, 4-34
  with COPY (L), 4-33 to 4-35 4-46, 8-33

E

E entity prefix (N), 5-4
EDIT (VAX/VMS terminal characteristic), D-6
EDIT system command, 2-5, 2-15, 2-17
EDIT/EDT (RSTS/E), E-1
EDIT/EDT (RSX-11M/M+), F-1
EDIT/EDT (VAX/VMS), D-1
Editing
  character, 2-10
  COMMAND (K) line, 3-58
  GOLD repeat number, 3-56
  journal file, 7-59 to 7-61
  key definitions, 7-26
  line, 2-10
  search string, 3-34
Editing keypad key numbers (LK201 keyboard), 6-15, 7-28
Editing mode, 2-1, 2-4
  See also keypad, line, nokeypad, hardcopy change
Editing session, 2-2, 2-4
  ending, 2-14, 2-16, 3-8 to 3-10, 4-10 to 4-11, 5-13 to 5-14
  recovering, 2-17
    INCLUDE (L), 2-17
    input file, 2-17
  starting, 2-14, 3-6 to 3-8, 4-13, 5-12 to 5-13
.EDT – default startup command file type, 7-1, D-3, E-6, F-2
EDT direction
  See direction
EDT macro
  See macro
EDT position
  See position
EDT prompt, 2-2
  See also prompt
/EDT qualifier, system, 2-5
/EDT qualifier (RSTS/E), E-3
/EDT qualifier (RSX-11M/M+), F-1
/EDT qualifier (VAX/VMS), D-1
EDT session
  See editing session
EDT system command, 2-5, 2-15
EDT CCL command (RSTS/E), E-6 to E-9
EDT MCR command (RSX-11M/M+), F-4 to F-7
EDTINI.EDT file, 7-1 to 7-2, D-3, D-7, E-3, E-6, F-2, F-5
EDTSYS.EDT file, 7-1 to 7-2, D-3, E-3, E-6, F-2, F-5
EIGHTBIT (VAX/VMS terminal characteristic), D-6
Eightbit graphics in terminals, E-9, F-9
EL (end of line) entity specifier (N), 5-4 to 5-6
End of buffer mark – [EOB], 2-9, 2-16, 3-7, 3-15, 4-7, 4-31, 4-36
End of line (EL) entity specifier (N), 5-4 to 5-6
End of line function
  See EOL (K)
End of page (EPAGE) entity specifier (N), 5-6, 5-45
End of paragraph (EPAR) entity specifier (N), 5-6, 5-45
End of range (ER) entity specifier (N), 5-6
End of sentence (ESEN) entity specifier (N), 5-6
End of word (EW) entity specifier (N), 5-5
END range specifier (L), 4-5, 4-7
Ending an editing session, 2-14, 2-16, 3-8 to 3-10, 4-10 to 4-11, 5-13 to 5-14
ENTER (K), 3-9, 3-32 to 3-34, 3-58, 7-17, 7-20, 7-21, 7-53, 8-89
  with COMMAND (K), 7-51
ENTER key, 3-9
Entering text
  See inserting text
Entity, 2-3, 2-9, 6-6
  character, 2-9
  line, 2-9
  page, 2-9, 6-9
  paragraph, 2-9, 6-8 to 6-9
  sentence, 2-9, 6-8
  SET ENTITY
    See SET ENTITY (L)
  word, 2-9, 6-7 to 6-8

Entity boundary, 5-6
Entity count specifier (N), 5-4
See also count specifier (N)
Entity specifier (N), 5-2, 5-3, 5-5 to 5-11, 8-92
B prefix (N), 5-4
BL (beginning of line), 5-5
BPAGE (beginning of page), 5-6
BPAR (beginning of paragraph), 5-6
BR (beginning of range), 5-6
BSEN (beginning of sentence), 5-6
BW (beginning of word), 5-5
C (character), 5-5
E prefix (N), 5-4
EL (end of line), 5-4, 5-5, 5-6
EPAGE (end of page), 5-6, 5-45
EPAR (end of paragraph), 5-6, 5-45
ER (end of range), 5-6
ESEN (end of sentence), 5-6
EW (end of word), 5-5
L (line), 5-5
NL (next line), 5-5
PAGE, 5-6
PAR (paragraph), 5-6
SEN (sentence), 5-5
SR (select range), 5-6, 5-29 to 5-30
string, 5-6, 5-9, 5-24, 5-26
with D (delete) (N), 5-20
V (vertical), 5-6, 5-9 to 5-10
with horizontal tab, 5-10
W (word), 5-5
with APPEND (N), 5-35 to 5-36, 8-8
with CHGC (change case) (N), 5-27, 8-21
with CHGL (change case lower) (N), 5-27, 8-22
with CHGU (change case upper) (N), 5-27, 8-23
with CUT (N), 5-31, 8-61
with D (delete) (N), 5-20 to 5-22, 8-62
with FILL (N), 5-40 to 5-41, 8-104
with "move" (N), 5-18 to 5-20, 8-129
with R (replace) (N), 5-24, 8-151
with TADJ (tab adjust) (N), 8-267
[EOB] (end of buffer mark), 2-9, 2-16, 3-7, 3-15, 4-7, 4-31, 4-36
EOL (K), 3-14, 7-21, 8-93
EPAGE (end of page) entity specifier (N), 5-6, 5-45
EPAR (end of paragraph) entity specifier (N), 5-6, 5-45
Equal sign (=)
buffer signal (L), 4-4, 4-28, 8-16
buffer signal (N), 5-3, 5-31, 5-32, 5-35, 8-16
current buffer indicator with SHOW BUFFER (L), 2-8, 6-12
Equal sign (=), (cont.)
with qualifier (RSTS/E), E-2
with qualifier (RSX-11M/M+), F-1
with qualifier (VAX/VMS), D-2
ER (end of range) entity specifier (N), 5-6
Errors in nokeypad multicommand line, 5-44
Escape character (<ESC>) in journal file, 7-58
Escape sequences in terminals, D-6, E-9, F-9
ESEN (end of sentence) entity specifier (N), 5-6
EW (end of word) entity specifier (N), 5-5
EX (exit to line mode) (N), 5-11, 5-13 to 5-14, 5-16, 7-50, 7-54, 7-76, 8-94
Exclamation point (!)
indicating comments in startup command files, 7-2
sentence boundary, 6-8
EXIT (L), 2-6 to 2-8, 2-15 to 2-17, 3-8 to 3-10, 4-1, 4-10 to 4-11, 4-13, 5-13 to 5-14, 8-95, D-1, D-2, D-8, E-3, F-7, F-8
from nokeypad mode, 5-15
tentative file, E-10
.TMP (temporary) file, D-8, F-9
with file specification specifier (L), 4-10
with /SAVE qualifier (L), 4-10, 4-46
with /SEQUENCE qualifier (L), 2-13, 4-10, 4-46, 7-82 to 7-83
EXT (extend) (N), 5-11, 5-15 to 5-16, 5-45, 7-54, 7-55, 7-76, 8-98
External file, 2-6, 2-16, 4-38 to 4-39

F

F12 key (LK201 keyboard), 2-14, 7-24, 7-30, 8-11
See also BACKSPACE (K)
F13 key (LK201 keyboard), 2-14, 7-24, 7-30, 8-126
See also LINEFEED (K)
<FF> – form feed, 3-15, 6-9
File, 2-2, 2-6, 2-15, 2-16
attributes, D-7, E-2, F-7
creating with EDT, 2-16, 3-7, 3-12, 4-13
EDTINI.EDT, 7-1 to 7-2, D-3, E-4, E-6, F-2, F-5
EDTSYS.EDT, 7-1 to 7-2, D-3, E-4, E-6, F-2, F-5
external, 2-6, 2-16, 4-38 to 4-39
HELP, 7-84 to 7-89, D-8, E-10, F-10
input, 2-2, 2-6, 2-16, 4-14, D-7, E-2, E-4, E-6 to E-9, F-5 to F-7
primary, 2-6, 7-82
sequence numbers with, 2-13
when recovering an editing session, 2-17

File, (cont.)
journal, 2-7, 2-17, 2-18, 4-11, 7-55 to 7-61, D-4, E-4 to E-9, F-3 to F-7
directory location, 2-18
with /JOURNAL qualifier (RSTS/E), E-4
with /JOURNAL qualifier (RSX-11M/M+), F-3, F-4
with /JOURNAL qualifier (VAX/VMS), D-4
with /NOOUTPUT qualifier (RSTS/E), E-5
with /NOOUTPUT qualifier (RSX-11M/M+), F-3
with /NOOUTPUT qualifier (VAX/VMS), D-4
with /READ_ONLY qualifier (VAX/VMS), D-5
with /RECOVER qualifier (RSTS/E), E-5
with /RECOVER qualifier (RSX-11M/M+), F-4
with /RECOVER qualifier (VAX/VMS), D-5
with /RO qualifier (RSTS/E), E-5
with /RO qualifier (RSX-11M/M+), F-4
LOGIN.COM, D-7
organization, D-8, E-3, F-7
output, 2-3 to 2-7, 3-8, 4-10, 4-40, D-1, D-4, D-7, E-2, E-4 to E-9, F-3 to F-8
primary, 2-7
sequence numbers with, 2-13
with PRINT (L), 7-83 to 7-84
with /READ_ONLY qualifier (VAX/VMS), D-5
with /RO qualifier (RSTS/E), E-5
with /RO qualifier (RSX-11M/M+), F-4
primary input, 7-82
protection, D-8, E-3, F-7
sequenced, 2-13, 7-81 to 7-82
startup command, 2-7, 5-42, 7-1 to 7-11, D-3, D-7, E-3, E-6 to E-9, F-2, F-5 to F-7
text, 2-2
version number (RSX-11M/M+), F-8
version number (VAX/VMS), D-1 to D-2
work, 7-81
File name, 2-7
journal, 2-17, D-4, D-5, E-2, E-4, E-5, F-3, F-4
File specification, 2-3, 2-6, 2-7, 2-15 to 2-17
prompt, 2-17
EDT system command, 2-15
File specification specifier (L)
with EXIT (L), 4-10, 8-95
with INCLUDE (L), 4-39, 8-120
with PRINT (L), 4-41, 8-146
File specification specifier (L), (cont.)
with SET COMMAND (L), 8-178
with SET HELP (L), 7-89, 8-184
with WRITE (L), 4-40, 8-292
File specification specifier (RSTS/E), E-1, E-6 to E-8
with qualifier, E-2 to E-5
File specification specifier (RSX-11M/M+), F-1, F-4 to F-7
with qualifier, F-2 to F-4
File specification specifier (VAX/VMS), D-1
with qualifier, D-2 to D-5
File type, 2-7
.BAK, E-2
.EDT, D-3, E-6, F-5
.JOU, 2-17, 2-18, D-5, E-4, E-6, F-3 to F-5
.TMP, D-8, F-9
FILL (K), 3-46, 3-48 to 3-49, 7-21, 8-99
with SET SCREEN (L), 3-48 to 3-49
with SET WRAP (L), 3-48 to 3-49
FILL (L), 4-2, 4-43 to 4-44, 8-102
with buffer specifier, 4-43 to 4-44
with range specifier, 4-43 to 4-44
with SET SCREEN (L), 4-43
with SET WRAP (L), 4-43
FILL (N), 5-11, 5-40 to 5-41, 8-104
with count specifier, 5-40 to 5-41
with entity specifier, 5-40 to 5-41
with SET SCREEN (L), 5-41
with SET WRAP (L), 5-40 to 5-41
with sign (+ | −) specifier, 5-40 to 5-41
Filling text
keypad mode, 3-36, 3-48 to 3-49
line mode, 4-43 to 4-44
nokeypad mode, 5-40 to 5-41
FIND (K), 3-14, 3-32 to 3-36, 7-21, 8-105
prompt – Search for:, 2-5, 3-32 to 3-36
FIND (L), 4-1, 4-16 to 4-18, 8-108
with buffer specifier, 4-30
with period (.) range specifier (L) and buffer specifier (L), 4-29
with range specifier, 4-30
Find key (LK201 keyboard), 7-24, 8-105
See also FIND (K)
FNDNXT (K), 3-14, 3-32, 3-35 to 3-36, 3-51, 7-21, 8-110
FOR n range specifier symbol (L), 4-5
Form feed character – CTRL/L, 2-9, 3-15,
page boundary, 3-15, 5-6, 6-9
with PRINT (L), 4-41
word boundary, 3-20, 6-7
Format, record, D-8, E-3, F-7
/FORMAT qualifier (RSTS/E), E-3, E-4

Formatting text
with FILL (K), 3-48 to 3-49
with FILL (L), 4-43 to 4-44
with FILL (N), 5-40 to 5-41
Forward direction, 2-10, 3-14, 3-37, 5-37 to 5-38
Fraction line numbers
See decimal fraction line numbers
Function, 3-2
alternate, 3-3, 3-6
help information, 3-10
preset, 3-1, 3-6
primary, 3-3
repeating, 3-6
FUNCTION key (LK201 keyboard), 3-3, 6-14, 7-15, 7-26
defining, 7-12 to 7-13
key numbers, 6-15, 7-28

G

GIGI (VK100) terminal, C-1
GOLD (K), 3-6, 7-21, 8-112
repeat function, 3-55 to 3-57
with minus sign, 3-55 to 3-57
GOLD CTRL (control) key sequence
defining, 7-12 to 7-14
in HELP file text, 7-88
GOLD DELETE key sequence
defining, 7-12 to 7-13
in HELP file text, 7-88
GOLD key, 3-2, 3-3, 3-6, 6-14
in HELP file text, 7-88
GOLD keyboard key sequence
defining, 7-12 to 7-14
GOLD keypad key sequence
defining, 7-12 to 7-13
GOLD/A (K), 7-22, 8-36
GOLD/D (K), 7-22, 8-39
GOLD/E (K), 7-22, 8-41
GOLD/R (K), 7-22, 8-51
GOLD/T (K), 7-22, 8-52
GOLD/U (K), 7-22, 8-54
GOLD/W (K), 7-22, 8-55
GOLD/Z (K), 7-22, 8-56

H

Hardcopy change mode, 2-2, 5-2, 7-75 to 7-80
prompt – C*, 7-76
shifting from line mode, 5-13, 7-49 to 7-50
Hardcopy terminal, 4-1, C-2, D-6, E-9, F-9
HELP (K), 3-6, 3-10, 7-21, 8-115
HELP (L), 4-1, 4-11 to 4-12, 8-116
for nokeypad information, 5-16
with wildcard character (*), 4-12
HELP (N), 5-11, 5-16, 8-117
HELP CHANGE (L), 5-16
ENTITIES, 5-16
SUBCOMMANDS, 5-16
HELP facility, 3-10 to 3-11, 4-11 to 4-12
HELP file, 7-84 to 7-89, D-8, E-10, F-10
HELP information
control key functions (K), 3-10
keypad mode, 3-6, 3-10
keyboard key functions, 3-10
line mode, 4-11 to 4-12
nokeypad mode, 5-16
screen display
VT100 terminal, 3-11
VT52 terminal, 3-11
HELP key, 3-3, 3-10
Help key (LK201 keyboard), 7-24, 8-115
See also HELP (K)
Horizontal tab, 3-53, 7-63
deleting, 3-53
with V (vertical) entity specifier (N), 5-10
word boundary, 3-20, 6-7

I

I (insert) (N), 5-11, 5-17 to 5-18, 5-45, 7-76, 8-118
in DEFINE KEY (L) definitions, 7-30
INCLUDE (L), 2-7, 4-2, 4-38 to 4-39, 7-82, 8-120, D-7, E-2, F-7
inserting sequence numbered file, 2-13
when recovering an editing session, 2-17
with buffer specifier, 4-39
with file specification specifier, 4-39
with range specifier, 4-39
Increment specifier (L)
with EXIT /SEQUENCE (L), 7-82, 8-95
with RESEQUENCE /SEQUENCE (L), 4-41 to 4-43, 7-82, 8-159
with /SEQUENCE qualifier (L), 7-81 to 7-83, 8-175
with WRITE /SEQUENCE (L), 7-82, 8-292
Indentation level, 7-66 to 7-75
Indenting text, 7-62 to 7-75
keypad mode, 3-36
Initial specifier (L)
with EXIT /SEQUENCE (L), 7-82, 8-95
with RESEQUENCE /SEQUENCE (L), 4-41 to 4-43, 7-82, 8-159
with /SEQUENCE qualifier (L), 7-81 to 7-83, 8-175
with WRITE /SEQUENCE (L), 7-82, 8-292

Initial tab stop, 7-63
Initialization file
See startup command file
Input file, 2-2, 2-6, 2-16, 4-14, D-7, E-2, E-4, E-6 to E-9, F-5 to F-7
primary, 2-6, 7-82
startup command, 2-7
when recovering an editing session, 2-17
INSERT (L), 4-1, 4-13 to 4-16, 4-36, 4-37, 4-45, 7-76, 8-121
single line form, 4-37
with buffer specifier, 4-37
with CTRL/Z, 4-37
with ;line-to-be-inserted, 4-37
with range specifier, 4-37
Insert Here key (LK201 keyboard), 7-24, 8-139
See also PASTE (K)
Insert state, 2-2, 4-13, 4-35, 4-37, 5-17 to 5-18, 5-24
Inserting text
keypad mode, 3-12 to 3-13
line mode, 4-13 to 4-16, 4-31, 4-35 to 4-36
line terminator, 2-14
nokeypad mode, 5-17 to 5-18
RETURN key, 2-14
sequence numbered file using INCLUDE (L), 2-13
Internal editing features in terminals, E-9, F-9
Interrupt operation – CTRL/C, 4-29
Interruption, system, 2-17, 2-18
Invoking EDT
See calling up EDT

J

.JOU – default journal file type, 2-17, 2-18, D-5, E-4, E-6, F-3 to F-5
directory location, 2-18
Journal facility, 2-17 to 2-18
Journal file, 2-7, 2-17, 2-18, 4-11, 7-55 to 7-61, E-6 to E-9, F-5 to F-7
directory location, 2-18
editing, 7-59 to 7-61
name, 2-17
type – .JOU default, 2-17 to 2-18
with /JOURNAL qualifier (RSTS/E), E-4
with /JOURNAL qualifier (RSX-11M/M+), F-3, F-4
with /JOURNAL qualifier (VAX/VMS), D-4
with /NOOUTPUT qualifier (RSTS/E), E-5
with /NOOUTPUT qualifier (RSX-11M/M+), F-3
Journal file, (cont.)
with /NOOUTPUT qualifier (VAX/VMS), D-4
with /READ_ONLY qualifier (VAX/VMS), D-5
with /RECOVER qualifier (RSTS/E), E-5
with /RECOVER qualifier (RSX-11M/M+), F-4
with /RECOVER qualifier (VAX/VMS), D-5
with /RO qualifier (RSTS/E), E-5
with /RO qualifier (RSX-11M/M+), F-4
/JOURNAL qualifier, 7-61
/JOURNAL qualifier (RSTS/E), E-4
with /RECOVER qualifier (RSTS/E), E-5
/JOURNAL qualifier (RSX-11M/M+), F-3
journal file with, F-4
with /RECOVER qualifier (RSX-11M/M+), F-4
/JOURNAL qualifier (VAX/VMS), D-3, D-4
with /RECOVER qualifier (VAX/VMS), D-4, D-5

K

Key
BACKSPACE, 3-2, 3-3
CTRL (control), 3-2, 3-3
DELETE, 3-2, 3-3
LINEFEED, 3-2, 3-3
TAB, 3-2, 3-3
Key definition, 3-6, 5-1, 5-2, 7-11 to 7-42
asterisk (*) with, 7-17
control characters, D-6, E-9, F-8
CTRL/@ as string delimiter, 7-18
CTRL/Z, 7-30
editing, 7-26
I (insert) (N), 7-30
parentheses [( )] in, 7-16 to 7-17
period (.) signal to process, 7-16, 7-27
prompts in, 7-17 to 7-19
question mark (?) prompt in, 7-17 to 7-19, 7-31
quotation marks (' ,"), 7-19, 7-31
with prompt text, 7-17 to 7-19
Key sequence, 3-2
control, 2-3, 2-9
Key-name specifier (L)
with DEFINE KEY (L), 7-26, 8-65
Keyboard, 3-2, 3-2 to 3-6
LK201, 3-3, 3-4, 7-28
main, 3-2
VT52 terminal, 3-4
VT100 terminal, 3-4

Keyboard key
  defining, 7-12 to 7-13
  help information, keypad mode, 3-10
Keypad, 2-14, 3-2 to 3-6
  key definition, 3-6
  LK201 editing, 7-28
  VT52 terminal, 3-5
  VT100 terminal, 3-5
Keypad HELP text, 7-87
Keypad key
  redefining, 7-12 to 7-13
Keypad key numbers, 6-14 to 6-15, 7-27 to 7-30
Keypad mode, 2-2, 3-1 to 3-61
  copying text, 3-36 to 3-46
  defining keys, 7-11 to 7-42
  deleting text, 3-18 to 3-30, 3-36 to 3-46
  filling text, 3-36, 3-48 to 3-49
  indenting text, 3-36
  inserting text, 3-12 to 3-13
  moving text, 3-36 to 3-46
  moving the cursor, 3-1, 3-14 to 3-17
  screen, 3-8
  shifting from line mode, 3-6, 3-59, 5-13, 7-49 to 7-50
  shifting from nokeypad mode, 7-55
  shifting to line mode, 3-9, 3-59 to 3-61, 7-50 to 7-52
  shifting to nokeypad mode, 7-52 to 7-54
  undeleting text, 3-18 to 3-30
  using line mode command from, 3-58 to 3-59
KS (KED substitute) (N), 5-11, 5-31, 5-36 to 5-37, 8-123

L

L (line) entity specifier (N), 5-5
LAST range specifier (L), 4-5, 4-8
LB:[1,2]EDTSYS.EDT file, F-2
LB:EDTSYS.EDT file, E-3
Left Arrow (K), 3-14, 3-15, 7-21, 8-124
Left Arrow (N), 8-124
Limits on searches, 6-11 to 6-12
Line, 2-10, 4-1
  adding in keypad mode, 3-30 to 3-31
  contiguous, 4-2
  current, 2-3, 4-4, 4-5, 4-16, 4-30, 4-32, 4-34 to 4-38, 4-49, 5-39
  multicommand (L), 4-45
  multicommand (N), 5-43 to 5-45
  renumbering, 4-41 to 4-44
LINE (K), 3-14, 7-21, 8-125
Line [L] entity specifier (N), 5-5
Line editing, 2-10
  See line mode
Line entity, 2-9
Line length, 4-44
  default, 4-43
Line mode, 2-2, 2-10, 4-1 to 4-49
  command, 4-2 to 4-3 abbreviation, 4-2 to 4-3
    with COMMAND (K), 3-9, 3-58 to 3-59
    with EXT (extend) (N), 5-15 to 5-16, 8-98
  copying text, 4-31, 4-33 to 4-35
  deleting text, 4-13 to 4-16, 4-31, 4-35 to 4-36
  filling text, 4-43 to 4-44
  inserting text, 4-13 to 4-16, 4-31, 4-35 to 4-36
  moving EDT's position, 4-16 to 4-18
  moving text, 4-31 to 4-33
  multicommand line, 4-45
  prompt – asterisk (*), 2-5, 2-16, 3-9, 3-59, 4-2, 5-14
  qualifiers, 4-3, 4-46 to 4-49
  replacing text, 4-35 to 4-36
  shifting from keypad mode, 3-9, 3-59 to 3-61, 7-50 to 7-52
  shifting from nokeypad mode, 5-14, 5-16, 7-54
  shifting to a change mode, 5-13
  shifting to hardcopy change mode, 7-49 to 7-50
  shifting to keypad mode, 3-6, 3-59, 7-49 to 7-50
  shifting to nokeypad mode, 7-49 to 7-50
  specifiers, 4-3
Line number, 2-4, 2-12, 4-1, 4-5 to 4-10, 4-41, 4-41 to 4-43, 7-80 to 7-84
  decimal fraction, 2-13, 7-81
Line number range specifier (L), 4-5, 4-6
Line ranges, 2-4, 2-13
  See also range specifier
Line terminator, 2-3, 2-14, 3-30 to 3-31
  adding in keypad mode, 3-30 to 3-31
  deleted by DEL L (K), 3-24
  in delete line buffer, 5-23
  in keypad search strings, 3-51 to 3-52
  in keypad substitute strings, 3-51 to 3-52
  inserting, 2-14
  sentence boundary, 6-8
  with FILL (K), 3-48
  with FILL (L), 4-43
  with UND L (K), 3-25
  with UNDL (N), 5-23
  word boundary, 3-20, 6-7
Line terminator symbol – <CR>, 7-76
;line-to-be-inserted
  with INSERT (L), 4-37, 8-121
  with REPLACE (L), 4-38, 8-157

Linefeed
word boundary, 3-20, 6-7
LINEFEED (K), 2-14, 3-18, 3-20 to 3-24, 7-14 to 7-15, 7-23, 7-25, 7-28, 8-126
LINEFEED (L), 2-14
LINEFEED (N), 2-14
LINEFEED key, 3-2, 3-3, 3-20, 6-14
LK201 editing keypad, 7-28
key numbers, 6-15, 7-28
LK201 FUNCTION key numbers, 6-15, 7-28
LK201 keyboard, 2-14, 3-3, 3-4, C-1
⟨X⟩key, 7-24, 7-29
Do key, 7-24
F12 key, 2-14, 7-24, 7-30
F13 key, 2-14, 7-24, 7-30
Find key, 7-24
FUNCTION keys, 6-14, 7-15, 7-26
defining, 7-12 to 7-13
Help key, 7-24
Insert Here key, 7-24
Next Screen key, 7-24
Prev Screen key, 7-24
Remove key, 7-24
Select key, 7-24
Location-1, 4-4
with COPY (L), 4-34 to 4-35
with MOVE (L), 4-32 to 4-33
Location-2, 4-4
with COPY (L), 4-34 to 4-35
with MOVE (L), 4-32 to 4-33
LOGIN.COM file, D-7

M

^M (CTRL/M), 3-9, 3-51, 3-58
See also RETURN
Macro, 5-42, 7-42 to 7-48
Macro-name specifier (L)
with DEFINE MACRO (L), 7-43 to 7-48, 8-68
MAIN buffer, 2-7, 2-8, 2-15, 2-16, 3-8, 4-4, 4-10, 4-28, 4-29, 4-31, 4-36, 6-12, 7-81, 7-82, 7-83
deleting contents with CLEAR (L), 2-8
Main keyboard, 3-2
Maximum number of characters on a command line – 4-45
MCR EDT (RSX-11M/M+), F-1, F-4
qualifiers, F-4 to F-7
specifiers, F-4 to F-7
Messages, A-1 to A-10
keypad mode, 3-8
Minus sign (–)
range specifier symbol (L), 4-5
sign specifier (N), 5-4
with string range specifier (L), 4-6
with TAB ADJUST (L), 7-74 to 7-75
Mode, 2-1
change, 2-1
hardcopy change, 5-2, 7-75 to 7-80
keypad, 3-1 to 3-61
line, 4-1 to 4-49
nokeypad, 5-1 to 5-45
screen, 2-1, C-1
MOVE (L), 4-1, 4-31, 4-31 to 4-33, 8-128
location-1, 4-32 to 4-33
location-2, 4-32 to 4-33
with buffer-1 specifier, 4-31 to 4-33
with buffer-2 specifier, 4-31 to 4-33
with /QUERY qualifier, 4-31, 4-46, 4-47 to 4-48
with range-1 specifier, 4-31 to 4-33
with range-2 specifier, 4-31 to 4-33
"move" (N), 5-8, 5-9, 5-10, 5-11, 5-18 to 5-20, 8-129
with count specifier, 5-18 to 5-20
with entity specifier, 5-18 to 5-20
with sign (+ | –) specifier, 5-18 to 5-20
with TOP (N), 5-39
Moving EDT's position in line mode, 4-16 to 4-18
Moving text
keypad mode, 3-36 to 3-46
line mode, 4-31 to 4-33
nokeypad mode, 5-31 to 5-37
Moving the cursor
keypad mode, 3-1, 3-14 to 3-17
nokeypad mode, 5-3, 5-18 to 5-20
Moving to another buffer, 4-4
Multicommand line (L), 4-45
semicolon (;) with, 4-45, 7-52, 7-54
Multicommand line (N), 5-1, 5-43 to 5-45

N

N (no), /QUERY qualifier response (L), 4-47 to 4-48
N (number) range specifier symbol (L), 4-5
n specifier (L)
with /BRIEF qualifier (L), 4-46, 8-15
with /DUPLICATE qualifier (L), 4-46, 8-88
with SET WRAP (L), 8-209
with TAB ADJUST (L), 7-74 to 7-75, 8-265
NEXT (L)
See SUBSTITUTE NEXT (L)
Next line (NL) entity specifier (N), 5-5
Next Screen key (LK201 keyboard), 7-24, 8-168
See also SECT (K)
NL (next line) entity specifier (N), 5-5
No –N, /QUERY qualifier response (L), 4-47 to 4-48

/NOCCL qualifier (RSTS/E), E-8
/NOCHAIN qualifier (RSTS/E), E-8
/NOCOMMAND qualifier (RSTS/E), E-3
/NOCOMMAND qualifier (RSX-11M/M+), F-2
/NOCOMMAND qualifier (VAX/VMS), D-3
/NOCREATE qualifier (RSTS/E), E-4
/NOCREATE qualifier (RSX-11M/M+), F-2
/NOCREATE qualifier (VAX/VMS), D-3
Node name, 2-7
/NOJOURNAL qualifier, 7-59
/NOJOURNAL qualifier (RSTS/E), E-4
/NOJOURNAL qualifier (RSX-11M/M+), F-3
/NOJOURNAL qualifier (VAX/VMS), D-3, D-4
Nokeypad command line
  error in, 5-44
  parentheses [( )] in, 5-44
  spaces in, 5-43
Nokeypad commands
  definitions for keypad editing keys, 7-15 to 7-17
  parentheses [( )] with, 5-42, 5-43 to 5-45
Nokeypad mode, 2-2, 5-1 to 5-45
  copying text, 5-31 to 5-37
  deleting text, 5-20 to 5-22
  ending an editing session, 5-13 to 5-14
  filling text, 5-40 to 5-41
  HELP information, 5-16
  inserting text, 5-17 to 5-18
  line mode commands from, 5-15 to 5-16
  moving text, 5-31 to 5-37
  moving the cursor, 5-3, 5-18 to 5-20
  shifting from keypad mode, 7-52 to 7-54
  shifting from line mode, 5-13, 7-49 to 7-50
  shifting to keypad mode, 7-55
  shifting to line mode, 5-14, 5-16, 7-54
  starting an editing session, 5-12 to 5-13
  undeleting text, 5-22 to 5-24
;nokeypad-command(s) specifier (L)
  with CHANGE (L), 5-13, 8-18
/NOOUTPUT qualifier (RSTS/E), E-5
/NOOUTPUT qualifier (RSX-11M/M+), F-3
/NOOUTPUT qualifier (VAX/VMS), D-3, D-4
  with journal file, D-4
/NOREAD_ONLY qualifier (VAX/VMS), D-3, D-5
/NORECOVER qualifier (RSTS/E), E-5
/NORECOVER qualifier (RSX-11M/M+), F-4
/NORECOVER qualifier (VAX/VMS), D-3, D-5
/NORO qualifier (RSTS/E), E-5
/NORO qualifier (RSX-11M/M+), F-4
/NOTYPE qualifier (L), 4-47, 8-131
  with SUBSTITUTE (L), 4-19, 4-46, 4-47, 8-255
<null> (L), 4-1, 4-16 to 4-18, 8-132
  with period (.) range specifier, 4-30
  with range specifier, 4-9, 4-30
Null character (^@) – CTRL/@, 5-41
  string delimiter in key definitions, 7-18
Number
  line, 2-4, 2-12, 4-1, 4-5 to 4-10, 4-41, 4-41 to 4-43, 7-80 to 7-84
    decimal fraction, 7-81
  sequence, 2-13, 7-81 to 7-82, 7-83
  See also decimal value of character
Number (n) range specifier symbol (L), 4-5
Number sign digit (# n) range specifier symbol (L), 4-5
Number specifier (L)
  with SET LINES (L), 8-186
  with SET TAB (L), 8-202
Number specifier (N), 5-3
  with ASC (ASCII) (N), 5-41 to 5-42, 8-9
Numbers for keypad keys, 6-14 to 6-15, 7-27 to 7-30

O

OPEN LINE (K), 3-30 to 3-31, 7-21, 8-135
Operating system, 2-2, 2-14, 2-16
Options for commands – enclosed in square brackets ([ ]), 4-3, 5-3
Output file, 2-3, 2-4, 2-5, 2-6, 2-7, 3-8, 4-10, 4-40, D-1, D-4, D-7, E-2, E-4, E-5, E-6 to E-9, F-3, F-4, F-5 to F-7, F-7, F-8
  primary, 2-7
  sequence numbers with, 2-13
  with PRINT (L), 7-83 to 7-84
  with /READ_ONLY qualifier (VAX/VMS), D-5
  with /RO qualifier (RSTS/E), E-5
  with /RO qualifier (RSX-11M/M+), F-4
/OUTPUT qualifier (RSTS/E), E-5
/OUTPUT qualifier (RSX-11M/M+), F-3
/OUTPUT qualifier (VAX/VMS), D-1, D-3, D-4

P

PAGE (K), 3-14, 3-15, 7-21, 8-137
Page boundary, 2-9, 6-9
  form feed, 3-15, 5-6
  with SET SEARCH BOUNDED (L), 6-12
Page entity, 2-9, 6-9
PAGE entity specifier (N), 5-6
PAR (paragraph) entity specifier (N), 5-6
Paragraph (PAR) entity specifier (N), 5-6

Paragraph boundary, 6-8
Paragraph entity, 2-9, 6-8 to 6-9
Parentheses [( )]
in key definitions, 7-16 to 7-17
with nokeypad commands, 5-42 to 5-45
PASTE (K), 3-36, 3-38 to 3-46, 7-21, 8-139
use of PASTE buffer with, 2-8
PASTE (N), 5-11, 5-31, 5-32 to 5-34, 8-143
with buffer specifier, 5-3, 5-32 to 5-34
with count specifier, 5-32 to 5-34
with KS (KED substitute) (N), 5-36 to 5-37
with PASTE buffer, 2-8, 5-3
PASTE buffer, 2-8, 3-37 to 3-46, 3-49 to 3-52, 4-31, 4-36, 5-31, 5-32, 5-35, 6-12
deleting contents with CLEAR (L), 2-8
with APPEND (K), 3-43 to 3-45
with APPEND (N), 5-3
with CUT (K), 3-38 to 3-39
with CUT (N), 5-3
with PASTE (K), 3-39 to 3-43
with PASTE (N), 5-3
with REPLACE (K), 3-46 to 3-47
with SUBS (K), 3-50 to 3-53
Percent sign (%), 4-19
with range specifier (L), 4-9, 4-16, 8-132
Period (.)
sentence boundary, 6-8
signal to process key definition, 7-16, 7-27
Period (.) range specifier (L), 4-5, 4-6
with FIND (L) and buffer specifier (L), 4-29
with <null> (L), 4-30
with TYPE (L), 4-29
Plus sign (+)
sign specifier (N), 5-4
range specifier symbol (L), 4-5
Position, 2-10
after line mode search, 2-11
after screen mode search, 2-11
Preset keypad function, 3-1, 3-6, 7-21 to 7-25
Prev Screen key (LK201 keyboard), 7-24, 8-168
See also SECT (K)
Primary function, 3-3
Primary input file, 2-6, 7-82
Primary output file, 2-7
PRINT (L), 2-7, 4-2, 4-38 to 4-39, 4-41, 7-83 to 7-84, 8-146, D-7, E-2, F-7
form feed, 4-41
tentative file, E-10
.TMP (temporary) file, D-8, F-9
with buffer specifier, 4-4, 4-28, 4-41
with file specification specifier, 4-41
with range specifier, 4-41
Printer port with VT100 terminals, C-1
Prompt, 2-2, 2-5
☞symbol for system prompt, 2-5
* (asterisk) – line mode, 2-5 2-16, 3-9, 3-59, 4-2, 5-14
? (question mark)
with /QUERY qualifier (L), 2-5, 4-47
in key definitions, 7-17 to 7-19, 7-31
',' (quotation marks) with text in key definitions, 7-17 to 7-19
C* – hardcopy change mode, 7-76
Command: – COMMAND (K), 2-5, 3-9, 3-58, 7-51 to 7-52
file specification, 2-17
EDT system command, 2-15
File: (RSTS/E), E-1
File? (RSX-11M/M+), F-1
_File: (VAX/VMS), D-1
in key definitions, 7-17 to 7-19
Search for: – FIND (K), 2-5, 3-32 to 3-36
search string (K), 2-5
system, 2-5
Prompt-type specifier (L)
with SET PROMPT (L), 8-192
with SHOW PROMPT (L), 8-228
Protection, file, D-8, E-3, F-7

Q

Q (quit), /QUERY qualifier response (L), 4-47 to 4-48
Qualifier, 2-4
EDIT system command, 2-5
/JOURNAL, 7-61
/NOJOURNAL, 7-59
/RECOVER, 7-61
Qualifier (L), 4-2, 4-3, 4-46 to 4-49
abbreviation, 4-3, 4-46
/BRIEF (L), 4-46 to 4-47, 8-15
with SUBSTITUTE (L), 4-19, 4-46, 4-46 to 4-47, 8-255
with TYPE (L), 4-16, 4-46 to 4-47, 8-277
/DUPLICATE (L), 8-88
with COPY (L), 4-33, 4-34 to 4-35, 4-46, 8-33
/NOTYPE (L), 8-131
with SUBSTITUTE (L), 4-19, 4-46, 4-47, 8-255
/QUERY (L), 4-47 to 4-48, 8-147
with buffer specifier (L), 4-47
with COPY (L), 4-33, 4-46, 4-47 to 4-48, 8-33
with DELETE (L), 4-36, 4-46, 4-47 to 4-48, 8-75

Qualifier (L), /QUERY (L), (cont.)
with MOVE (L), 4-31, 4-46, 4-47 to 4-48, 8-128
with range specifier (L), 4-47
with SUBSTITUTE (L), 4-19, 4-46, 4-47 to 4-48, 8-255
/SAVE (L), 8-167
with EXIT (L), 4-10, 4-46, 8-95
with QUIT (L), 4-10, 4-46, 7-55, 8-149
/SEQUENCE (L), 2-13, 7-82 to 7-83, 8-175, D-8, E-3, F-7
with EXIT (L), 4-10, 4-46, 7-82 to 7-83, 8-95
with RESEQUENCE (L), 4-46, 7-82 to 7-83, 8-159
with WRITE (L), 4-46, 7-82 to 7-83, 8-292
slash (/) indicator (L), 4-3
/STAY (L), 4-49, 8-249
with TYPE (L), 4-16, 4-46, 4-49, 8-277
Qualifier (RSTS/E)
/CCL, E-8
CCL EDT, E-6 to E-9
/CHAIN, E-8
/COMMAND, E-3
/CREATE, E-4, E-8
DCL, E-1, E-3 to E-6
/FORMAT, E-3, E-4
/JOURNAL, E-4
with /RECOVER qualifier (RSTS/E), E-5
/NOCCL, E-8
/NOCHAIN, E-8
/NOCOMMAND, E-3, E-8
/NOCREATE, E-4, E-8
/NOJOURNAL, E-4, E-8
/NOOUTPUT, E-5, E-8
/NORECOVER, E-5, E-8
/NORO, E-5, E-8
/NOSTREAM, E-8
/NOVARIABLE, E-8
/OUTPUT, E-5
/RECOVER, E-5, E-8
with /JOURNAL qualifier (RSTS/E), E-4
/RO, E-5, E-8
/STREAM, E-8
/VARIABLE, E-8
Qualifier (RSX-11M/M+)
/COMMAND, F-2
/CREATE, F-2, F-7
DCL EDIT/EDT, F-1 to F-4
/JOURNAL, F-3
with /RECOVER qualifier (RSX-11M/M+), F-4
Qualifier (RSX-11M/M+), (cont.)
MCR EDIT/EDT, F-4 to F-7
/NOCOMMAND, F-2, F-7
/NOCREATE, F-2, F-7
/NOJOURNAL, F-3, F-7
/NOOUTPUT, F-3, F-7
/NORECOVER, F-4, F-7
/NORO, F-4, F-7
/OUTPUT, F-3
/RECOVER, F-4, F-7
with /JOURNAL qualifier (RSX-11M/M+), F-3
/RO, F-4, F-7
with /JOURNAL qualifier (RSX-11M/M+), F-3
Qualifier (VAX/VMS), D-1, D-2 to D-5
/COMMAND, D-3
/CREATE, D-3
/EDT, D-1
/JOURNAL, D-3, D-4
with /RECOVER qualifier (VAX/VMS), D-5
/NOCOMMAND, D-3
/NOCREATE, D-3
/NOJOURNAL, D-3, D-4
/NOOUTPUT, D-3, D-4
with journal file, D-4
/NOREAD_ONLY, D-3, D-5
/NORECOVER, D-3, D-5
/OUTPUT, D-1, D-3, D-4
/READ_ONLY, D-3, D-5
/RECOVER, D-3, D-5
with /JOURNAL qualifier (VAX/VMS), D-4
with colon (:), D-2
with equal sign (=), D-2
with file specification specifier, D-2
Qualifier indicator – slash (/) (L), 4-3, 4-46
/QUERY qualifier (L), 4-47 to 4-48, 8-147
prompt – question mark (?) (L), 2-5, 4-47 to 4-48
responses
A (all) (L), 4-47 to 4-48
N (no) (L), 4-47 to 4-48
Q (quit) (L), 4-47 to 4-48
Y (yes) (L), 4-47 to 4-48
with buffer specifier (L), 4-47
with COPY (L), 4-33, 4-46 to 4-48, 8-33
with DELETE (L), 4-36, 4-46 to 4-48, 8-75
with MOVE (L), 4-31, 4-46 to 4-48, 8-128
with range specifier (L), 4-47
with SUBSTITUTE (L), 4-19, 4-46 to 4-48, 8-255

Question mark (?)
prompt for /QUERY qualifier (L), 2-5, 4-47 to 4-48
prompt in key definitions, 7-17 to 7-19, 7-31
sentence boundary, 6-8
QUIT (L), 2-6, 2-8, 2-15 to 2-17, 3-8, 3-9, 4-1, 4-10 to 4-11, 5-13 to 5-14, 7-55, 8-149
with /SAVE qualifier (L), 4-10, 4-46, 7-55
QUIT (N), 5-11, 5-13 to 5-14, 8-150
Quit –Q, /QUERY qualifier response (L), 4-47 to 4-48
Quotation marks (' , ")
delimiter character, 2-11, 4-2
in key definitions, 7-19, 7-31
string delimiter, 4-6, 5-30
with prompt text in key definitions, 7-17 to 7-19
with SET ENTITY (L), 6-6
with string specifier (L), 6-1

R

R (replace) (N), 5-11, 5-24, 5-45, 8-151
with count specifier, 5-24
with entity specifier, 5-24
with sign (+ | –) specifier, 5-24
Range, 2-4, 2-13
select, 2-3, 2-12, 3-36 to 3-49, 4-9, 4-40, 4-43, 5-29 to 5-30
active, 3-38, 3-46, 5-29 to 5-30
cancelling, 3-37, 5-30
changing case of letters, 3-46 to 3-48, 5-27 to 5-28
creating, 3-36 to 3-37, 5-29 to 5-30
specifying two or more in a single command, 2-13
Range specifier (L), 4-2, 4-4 to 4-10, 4-14, 8-152
BEFORE, 4-5, 4-8
BEGIN, 4-5, 4-7
END, 4-5, 4-7
LAST, 4-5, 4-8
line number, 4-5, 4-6
percent sign (%) with, 4-9, 4-16
period (.), 4-5, 4-6
with <null> (L), 4-30
with TYPE (L), 4-29
REST, 4-5, 4-8
SELECT, 2-12, 4-5, 4-9, 4-40
string, 4-5 to 4-7
minus sign (–) with, 4-6
Range specifier (L), (cont.)
symbol
ALL, 4-5, 4-6
AND, 4-5
colon (:), 4-5
comma (,), 4-5
FOR n, 4-5
minus sign (–), 4-5
number (n), 4-5
number sign digit (# n), 4-5
plus sign (+), 4-5
THRU, 4-5
WHOLE, 4-5, 4-8
with SUBSTITUTE (L), 4-20
with CHANGE (L), 5-13, 8-18
with DELETE (L), 4-36, 8-75
with FILL (L), 4-43 to 4-44, 8-102
with FIND (L), 4-16 to 4-18, 4-30, 8-108
with INCLUDE (L), 4-39, 8-120
with INSERT (L), 4-37, 8-121
with <null> (L), 4-9, 4-16 to 4-18, 4-30, 8-132
with PRINT (L), 4-41, 8-146
with /QUERY qualifier (L), 4-47
with REPLACE (L), 4-35 to 4-36, 4-38, 8-157
with RESEQUENCE (L), 4-41 to 4-43, 8-159
with SUBSTITUTE (L), 4-19 to 4-21, 4-22 to 4-23, 8-255
with SUBSTITUTE NEXT (L), 4-19, 4-21 to 4-23
with TAB ADJUST (L), 7-74 to 7-75, 8-265
with TYPE (L), 4-16 to 4-18, 4-30, 8-277
with WRITE (L), 4-40, 8-292
Range-1 specifier (L), 4-4
with COPY (L), 4-33 to 4-35, 8-33
with MOVE (L), 4-31 to 4-33, 8-128
Range-2 specifier (L), 4-4
with COPY (L), 4-33 to 4-35, 8-33
with MOVE (L), 4-31 to 4-33, 8-128
/READ_ONLY qualifier (VAX/VMS), D-3, D-5
Record attribute, D-8, E-3, F-7
Record format, D-8, E-3, F-7
VFC, 7-82, 7-83
/RECOVER qualifier, 2-17, 2-18, 7-61
/RECOVER qualifier (RSTS/E), E-5
with /JOURNAL qualifier (RSTS/E), E-4
/RECOVER qualifier (RSX-11M/M+), F-4
with /JOURNAL qualifier (RSX-11M/M+), F-3
/RECOVER qualifier (VAX/VMS), D-3, D-5
with /JOURNAL qualifier (VAX/VMS), D-4

Recovering an editing session, 2-17
INCLUDE (L), 2-17
input file, 2-17
Redefining keys
See defining keys
REF (refresh) (N), 5-11, 5-38, 8-154
Refreshing the screen
keypad mode, 3-12
nokeypad mode, 5-38
Remove key (LK201 keyboard), 7-24, 8-59
See also CUT (K)
Renumbering lines, 2-13, 4-41 to 4-44
Repeat count limit, 3-55
count specifier (N), 5-3
/DUPLICATE qualifier (L), 4-34
Repeat count specifier (N), 5-4
See also count specifier (N)
Repeat function, keypad mode, 3-55 to 3-57
Repeat number, editing, 3-56
Repeating a keypad function, 3-6, 3-55 to 3-57, 6-2
REPLACE (K), 3-36, 3-45 to 3-46, 7-21, 8-155
with DELETE buffer, 6-13
REPLACE (L), 4-1, 4-31, 4-35 to 4-36, 4-38, 4-45, 8-157
single line form, 4-38
with buffer specifier, 4-35 to 4-36, 4-38
with CTRL/Z, 4-35 to 4-36
with ;line-to-be-inserted, 4-38
with range specifier, 4-35 to 4-36, 4-38
Replacing text
keypad mode, 3-46 to 3-47
line mode, 4-35 to 4-36
nokeypad mode, 5-24
See also substituting
RESEQUENCE (L), 2-13, 4-2, 4-41 to 4-43, 7-83, 8-159
with buffer specifier, 4-41 to 4-43
with range specifier, 4-41 to 4-43
with /SEQUENCE qualifier (L) 4-41 to 4-43, 4-46, 7-82 to 7-83
RESET (K), 3-36, 3-37, 7-21, 8-160
editing key definitions, 7-26
REST range specifier (L), 4-5, 4-8
Restoring text
See undeleting
RETURN (K), 3-13, 3-25, 3-30 to 3-31, 3-51, 7-14 to 7-15, 7-17, 7-23, 7-25, 7-28, 8-49, 8-162
RETURN key, 2-14, 3-9, 3-58, 4-13, 4-36, 5-1, 5-3, 5-17, 5-24
inserting text, 2-14
Right Arrow (K), 3-14, 3-15, 7-21, 8-164
Right Arrow (N), 8-164
/RO qualifier (RSTS/E), E-5
/RO qualifier (RSX-11M/M+), F-4
with /JOURNAL qualifier (RSX-11M/M+), F-3
RSTS/E, 2-15, 3-10, E-1 to E-13
RSX-11M/M-PLUS, 2-15, F-1 to F-18

S

S (substitute) (N), 5-11, 5-24 to 5-26, 8-165
with count specifier, 5-24 to 5-26
with sign (+ | –) specifier, 5-24 to 5-26
with string specifier, 5-4
with string-1 specifier, 5-24 to 5-26
with string-2 specifier, 5-24 to 5-26
/SAVE qualifier (L), 4-11, 8-167
with EXIT (L), 4-10, 4-46, 8-95
with QUIT (L), 4-10, 4-46, 7-55, 8-149
Scope (RSTS/E), E-10
Screen
keypad mode, 3-8
refreshing in keypad mode, 3-12
refreshing in nokeypad mode, 5-38
Screen display, 5-38 to 5-39
shifting horizontally, 5-39
Screen mode, 2-1, 2-10, 5-1, C-1
searching, 2-12. 6-10
Screen terminal, 4-1, C-1
Screen width, 5-40 to 5-41, C-1, D-7, E-9, F-9
VK100 Terminal, C-1
Scrolling regions in terminals, D-6, E-9, F-9
Search, 2-10, 2-12, 3-32 to 3-36, 6-9 to 6-12
cancelling, 3-34
direction, 3-32 to 3-34
screen mode, 2-12
Search buffer, 2-8, 3-32 to 3-36, 3-49 to 3-52, 4-19, 5-25, 5-26, 5-30, 5-40, 6-12
deleting the contents – CLSS (N), 3-35
Search for: FIND (K) prompt, 2-5
Search string, 2-10, 2-12, 3-32 to 3-36, 3-49 to 3-52, 4-19, 4-22, 5-24, 5-26 to 5-27, 6-9 to 6-12
case of letters, 2-12, 6-10
changing, 3-46 to 3-48, 5-27 to 5-28
CLSS (clear search string) (N), 5-40
current, 3-35
diacritical marks in, 2-12
editing, 3-34

Searching
See also SET SEARCH (L)
SECT (K), 3-14, 3-15, 7-21, 8-168
+ Sect (K) (LK201 keyboard), 8-168
See also SECT (K)
– Sect (K) (LK201 keyboard), 8-168
See also SECT (K)
Section function
See SECT (K)
SEL (select) (N), 2-12, 4-5, 4-40, 4-43, 5-11, 5-29 to 5-30, 8-170
SELECT (K), 2-12, 3-36 to 3-49, 4-5, 4-40, 4-43, 7-21, 8-172
Select (LK201 keyboard), 8-172
See also SELECT (K)
Select key (LK201 keyboard), 7-24
Select range, 2-3, 2-12, 3-36 to to 3-49, 4-9, 4-40, 4-43, 5-29 to 5-30
active, 3-38, 3-46, 5-29, 5-30
cancelling, 3-37, 5-30
changing case of letters, 3-46 to 3-48, 5-27 to 5-28
creating, 3-36 to 3-37, 5-29 to 5-30
See also SELECT range specifier (L)
Select range (SR) entity specifier (N), 5-6, 5-29 to 5-30
SELECT range specifier (L), 2-12, 4-5, 4-9, 4-40, 5-30
Semicolon (;)
with INSERT (L) line-to-be-inserted, 4-37
with multicommand line (L), 4-45, 7-52, 7-54
with REPLACE (L) line-to-be-inserted, 4-38
SEN (sentence) entity specifier (N), 5-5
Sentence boundary, 6-8
Sentence entity, 2-9, 6-8
Sentence (SEN) entity specifier (N), 5-5
Sequence number, 2-13, 7-81 to 7-83
Sequence numbered file, 2-13 7-81 to 7-82
inserting using INCLUDE (L), 2-13
/SEQUENCE qualifier (L), 2-13, 7-81 to 7-83, 8-175, D-8, E-3, F-7
with EXIT (L), 2-13, 4-10, 4-46, 7-82 to 7-83, 8-95
with increment specifier (L), 4-41 to 4-43, 7-81 to 7-83
with initial specifier (L), 4-41 to 4-43, 7-81 to 7-83
with RESEQUENCE (L), 4-41 to 4-43, 4-46, 7-82 to 7-83, 8-159
with WRITE (L), 2-13, 4-40, 4-46, 7-82 to 7-83, 8-292
SET (L), 4-2, 6-1 to 6-16
with string specifier, 6-1
SET AUTOREPEAT (L), 6-2, 8-176, C-1
SET CASE (L), 6-2, 8-177
LOWER, 6-2, 8-177
NONE, 6-2, 8-177
UPPER, 6-2, 8-177
SET COMMAND (L), 6-2, 7-10 to 7-11, 8-178
SET CURSOR (L), 6-2, 8-180, C-3
SET ENTITY (L), 2-9, 6-2, 6-6 to 6-9, 8-181
ASC (ASCII) (N) with, 6-6
control characters with, 6-6
PAGE, 6-2, 6-9, 8-181
PARAGRAPH, 6-2, 6-8 to 6-9, 8-181
quotation marks (',") with, 6-6
SENTENCE, 6-2, 6-8, 8-181
SPECINS (K) with, 6-6
string specifier (L) with, 6-6
WORD, 3-23, 6-2, 6-7 to 6-8, 8-181
SET FNF (L), 6-2, 8-183
SET HELP (L), 4-12, 6-3, 7-89, 8-184
with file specification specifier, 7-89
SET KEYPAD (L), 6-3, 7-49, 7-52, 7-55, 8-185
SET LINES (L), 6-3, 8-186, C-3
SET MODE (L), 6-3, 8-188
CHANGE, 6-3, 8-188
LINE, 6-3, 8-188
SET NOAUTOREPEAT (L), 6-2, 8-176, C-1
SET NOFNF (L), 6-2, 8-183
SET NOKEYPAD (L), 5-12, 6-3, 7-49, 7-52, 8-185
SET NONUMBERS (L), 4-44, 4-45, 6-3, 7-80, 8-189
SET NOQUIET (L), 6-4, 8-194
SET NOREPEAT (L), 3-55, 3-57, 6-4, 8-195
SET NOSUMMARY (L), 6-4, 8-201
SET NOTAB (L), 3-52, 6-5, 7-63, 8-202
SET NOTRUNCATE (L), 6-5, 8-206
SET NOVERIFY (L), 6-5, 8-207
SET NOWRAP (L), 4-43, 6-5, 8-209
with FILL (K), 3-48 to 3-49
with FILL (L), 4-43
with FILL (N), 5-41
SET NUMBERS (L), 6-3, 7-80, 8-189
SET PARAGRAPH (L), 6-3, 8-190
NOWPS, 6-3, 8-190
WPS, 6-3, 8-190
SET PROMPT (L), 6-4, 8-192
SET QUIET (L), 6-4, 8-194
SET REPEAT (L), 3-55, 3-57, 6-4, 8-195
SET SCREEN (L), 6-4, 8-196, C-2, D-7, E-9, F-9
with FILL (K), 3-48 to 3-49
with FILL (L), 4-43
with FILL (N), 5-41

SET SEARCH (L), 2-12, 6-4, 6-9 to 6-12, 8-198
BEGIN, 6-4, 6-11, 8-198
BOUNDED, 6-4, 8-198
page boundaries with, 6-12
CASE INSENSITIVE, 6-4, 6-11, 8-198
DIACRITICAL INSENSITIVE, 6-4, 6-11, 8-198
END, 6-4, 6-11, 8-198
EXACT, 6-4, 6-10, 8-198
GENERAL, 6-4, 6-10, 8-198
UNBOUNDED, 6-4, 6-12, 8-198
WPS, 6-4, 6-10 to 6-11, 8-198
SET SUMMARY (L), 6-4, 8-201
SET TAB (L), 6-5, 7-62, 7-63, 7-66 to 7-75, 8-202
value, 7-63, 7-66 to 7-75
SET TERMINAL (L), 6-5, 8-203, C-2, D-7
EDIT, 6-5, 8-203, C-2
EIGHTBIT, 6-5, 8-203, C-2
HCPY, 6-5, 8-203, C-2
NOEDIT, 6-5, 8-203, C-2
NOEIGHTBIT, 6-5, 8-203, C-2
NOSCROLL, 6-5, 8-203, C-2
SCROLL, 6-5, 8-203, C-2
VT100, 6-5, 8-203, C-2
VT52, 6-5, 8-203, C-2
SET TERMINAL (RSTS/E), E-9
SET TEXT (L), 6-5, 8-205
END, 6-5, 8-205
PAGE, 6-5, 8-205
SET TRUNCATE (L), 6-5, 8-206
SET VERIFY (L), 6-5, 8-207
SET WORD (L), 6-5, 8-208
DELIMITER, 6-5, 8-208
NODELIMITER, 3-24, 6-5, 8-208
SET WRAP (L), 6-5, 8-209
with FILL (K), 3-48 to 3-49
with FILL (L), 4-43
with FILL (N), 5-40 to 5-41
Shifting modes, 7-49 to 7-55
keypad to line mode, 3-9, 3-59 to 3-61, 7-50 to 7-52
keypad to nokeypad mode, 7-52 to 7-54
line to change mode, 5-13
line to hardcopy change mode, 7-49 to 7-50
line to keypad mode, 3-6, 3-59, 7-49 to 7-50
line to nokeypad mode, 7-49 to 7-50
nokeypad to keypad mode, 7-55
nokeypad to line mode, 5-14, 5-16, 7-54
Shifting the screen display horizontally, 5-39
SHL (shift left) (N), 5-11, 5-39, 8-211
with count specifier, 5-39
SHOW (L), 4-2, 4-45, 6-1 to 6-16
SHOW AUTOREPEAT (L), 6-2, 8-212
SHOW BUFFER (L), 2-8, 3-37, 4-29 to 4-31, 5-3, 6-2, 6-12 to 6-14, 8-213
asterisk (*) in, 6-12
current buffer indicator – equal sign (=), 2-8, 6-12
SHOW CASE (L), 6-2, 8-214
SHOW COMMAND (L), 6-2, 7-10 to 7-11, 8-215
SHOW CURSOR (L), 6-2, 8-216
SHOW ENTITY (L), 6-2, 6-6 to 6-9, 8-217
PAGE, 6-2, 8-217
PARAGRAPH, 6-2, 8-217
SENTENCE, 6-2, 8-217
WORD, 3-24, 6-2, 6-7, 8-217
SHOW FILES (L), 6-2, 8-218
SHOW FNF (L), 6-2, 8-219
SHOW HELP (L), 4-12, 6-3, 7-89, 8-220
SHOW KEY (L), 6-3, 6-14 to 6-16, 7-16, 7-27, 7-28, 8-221
keypad key numbers, 6-15, 7-27
SHOW KEYPAD (L), 6-3, 8-223
SHOW LINES (L), 6-3, 8-224
SHOW MODE (L), 6-3, 8-225
SHOW NUMBERS (L), 6-3, 8-226
SHOW PARAGRAPH (L), 6-3, 8-227
SHOW PROMPT (L), 6-4, 8-228
SHOW QUIET (L), 6-4, 8-229
SHOW REPEAT (L), 6-4, 8-230
SHOW SCREEN (L), 6-4, 8-231, C-2
SHOW SEARCH (L), 6-9 to 6-12, 8-232
SHOW SUMMARY (L), 6-4, 8-234
SHOW TAB (L), 6-5, 7-66, 7-67, 7-73, 8-235
SHOW TERMINAL (L), 6-5, 7-49, 8-237, C-2
SHOW TEXT (L), 6-5, 8-238
END, 6-5, 8-238
PAGE, 6-5, 8-238
SHOW TRUNCATE (L), 6-5, 8-239
SHOW VERIFY (L), 6-5, 8-240
SHOW VERSION (L), 6-5, 8-241
SHOW WORD (L), 6-5, 8-242
SHOW WRAP (L), 6-5, 8-243
SHR (shift right) (N), 5-11, 5-39, 8-244
with count specifier, 5-39
Sign (+ | –) specifier (N), 5-2 to 5-4, 5-6
with APPEND (N), 5-35 to 5-36, 8-8
with CHGC (change case) (N), 5-27, 8-21
with CHGL (change case lower) (N), 5-27, 8-22
with CHGU (change case upper) (N), 5-27, 8-23

Sign (+ | –) specifier (N), (cont.)
with CUT (N), 5-31, 8-61
with D (delete) (N), 5-20 to 5-22, 8-62
with FILL (N), 5-40 to 5-41, 8-104
with "move" (N), 5-18 to 5-20, 8-129
with R (replace) (N), 5-24, 8-151
with S (substitute) (N), 5-24 to 5-26, 8-165
with SN (substitute next) (N), 5-26 to 5-27, 8-245
with SSEL (search and select) (N), 5-29, 5-30, 8-248
with TADJ (tab adjust) (N), 8-267
Single quotation mark (')
delimiter character, 2-11, 4-2, 4-6, 5-4, 5-6, 5-30, 6-1, 6-6, 7-14, 7-17 to 7-19, 7-26, 7-27
in key definitions, 7-31
with prompt text in key definitions, 7-17 to 7-19
Slash (/), 4-47
qualifier indicator (L), 4-3, 4-46
SUBSTITUTE (L) delimiter, 4-19
SN (substitute next) (N), 5-11, 5-24, 5-26 to 5-27, 8-245
with count specifier, 5-26 to 5-27
with sign (+ | –) specifier, 5-26 to 5-27
Space
in delete word buffer, 5-23
in nokeypad command syntax, 5-3, 5-43
sentence boundary, 6-8
with FILL (L), 4-43
word boundary, 3-20, 6-7
Specification
See directory specification
See file specification
Specifier, 2-4, 4-2 to 4-3, 5-3 to 5-10, 8-18
bottom (L) with SET CURSOR (L), 8-180
buffer (L), 2-8, 4-4, 4-28, 4-30, 8-16
with CHANGE (L), 5-13, 8-18
with CLEAR (L), 4-4, 4-28, 4-30 to 4-31, 8-28
with DELETE (L), 4-36, 8-75
with FILL (L), 4-43 to 4-44, 8-102
with FIND (L), 4-16 to 4-18, 4-30, 8-108
period (.) range specifier (L), 4-29
with INCLUDE (L), 4-39, 8-120
with INSERT (L), 4-37, 8-121
with <null> (L), 4-16 to 4-18, 4-30, 8-132
with PRINT (L), 4-4, 4-28, 4-41, 8-146
with /QUERY qualifier (L), 4-47
with REPLACE (L), 4-35 to 4-36, 4-38, 8-157
with RESEQUENCE (L), 4-41 to 4-43, 8-159
Specifier, buffer (L), (cont.)
with SUBSTITUTE (L), 4-19, 4-19 to 4-21, 4-22 to 4-23, 8-255
with SUBSTITUTE NEXT (L), 4-19, 4-21 to 4-23
with TAB ADJUST (L), 7-74 to 7-75, 8-265
with TYPE (L), 4-16 to 4-18, 4-30, 8-277
with WRITE (L), 4-4, 4-28, 4-40, 8-292
buffer (N), 2-8, 5-2, 5-3, 8-16
with APPEND (N), 5-3, 5-35 to 5-36, 8-8
with CUT (N), 5-3, 5-31, 8-61
with PASTE (N), 5-3, 5-32 to 5-34, 8-143
buffer-1 (L), 4-4
with COPY (L), 4-33 to 4-35, 8-33
with MOVE (L), 4-31 to 4-33, 8-128
buffer-2 (L), 4-4
with COPY (L), 4-33 to 4-35, 8-33
with MOVE (L), 4-31 to 4-33, 8-128
character (N), 5-3
with circumflex (^) (N), 5-42 to 5-43, 8-27
count (N), 5-2, 5-3 to 5-4, 8-35
with APPEND (N), 5-35 to 5-36, 8-8
with ASC (ASCII) (N), 5-42, 8-9
with CHGC (change case) (N), 5-27, 8-21
with CHGL (change case lower) (N), 5-27, 8-22
with CHGU (change case upper) (N), 5-27, 8-23
with circumflex (^) (N), 5-42 to 5-43, 8-27
with CUT (N), 5-31, 8-61
with D (delete) (N), 5-20 to 5-22, 8-62
with FILL (N), 5-40 to 5-41, 8-104
with "move" (N), 5-18 to 5-20, 8-129
with PASTE (N), 5-32 to 5-34, 8-143
with R (replace) (N), 5-24, 8-151
with S (substitute) (N), 5-24 to 5-26, 8-165
with SHL (shift left) (N), 5-39, 8-211
with SHR (shift right) (N), 5-39, 8-244
with SN (substitute next) (N), 5-26 to 5-27, 8-245
with TAB (N), 8-263
with TADJ (tab adjust) (N), 8-267
with TD (tab decrement) (N), 8-270
with TI (tab increment) (N), 8-273
with UNDC (undelete character) (N), 5-22 to 5-24, 8-280
with UNDL (undelete line) (N), 5-22 to 5-24, 8-284
with UNDW (undelete word) (N), 5-22 to 5-24, 8-288
entity (N), 5-2, 5-3, 5-5 to 5-11, 8-92
BL (beginning of line), 5-5
BPAGE (beginning of page), 5-6
BPAR (beginning of paragraph), 5-6
BR (beginning of range), 5-6

Specifier, entity (N), (cont.)
BSEN (beginning of sentence), 5-6
BW (beginning of word), 5-5
C (character), 5-5
EL (end of line), 5-4, 5-5, 5-6
EPAGE (end of page), 5-6, 5-45
EPAR (end of paragraph), 5-6, 5-45
ER (end of range), 5-6
ESEN (end of sentence), 5-6
EW (end of word), 5-5
L (line), 5-5
NL (next line), 5-5
PAGE, 5-6
PAR (paragraph), 5-6
SEN (sentence), 5-5
SR (select range), 5-6, 5-29 to 5-30
string, 5-6, 5-9, 5-24, 5-26
with D (delete) (N), 5-20
V (vertical), 5-6, 5-9 to 5-10
with horizontal tab, 5-10
W (word), 5-5
with APPEND (N), 5-35 to 5-36, 8-8
with CHGC (change case) (N), 5-27, 8-21
with CHGL (change case lower) (N), 5-27, 8-22
with CHGU (change case upper) (N), 5-27, 8-23
with CUT (N), 5-31, 8-61
with D (delete) (N), 5-20 to 5-22, 8-62
with FILL (N), 5-40 to 5-41, 8-104
with "move" (N), 5-18 to 5-20, 8-129
with R (replace) (N), 5-24, 8-151
with TADJ (tab adjust) (N), 8-267
entity count (N), 5-4
file specification (L)
with EXIT (L), 4-10, 8-95
with INCLUDE (L), 4-39, 8-120
with PRINT (L), 4-41, 8-146
with SET COMMAND (L), 8-178
with SET HELP (L), 7-89, 8-184
with WRITE (L), 4-40, 8-292
file specification (RSTS/E), E-1, E-6
file specification (RSX-M/M+), F-1, F-4
file specification (VAX/VMS), D-1
increment (L)
with EXIT /SEQUENCE (L), 7-82, 8-95
with RESEQUENCE (L), 4-41 to 4-43
with RESEQUENCE /SEQUENCE (L), 4-41 to 4-43, 7-82, 8-159
with /SEQUENCE qualifier (L), 7-81 to 7-83, 8-175
with WRITE /SEQUENCE (L), 7-82, 8-292
initial (L)
with EXIT /SEQUENCE (L), 7-82, 8-95
with RESEQUENCE (L), 4-41 to 4-43
with RESEQUENCE /SEQUENCE (L), 4-41 to 4-43, 7-82, 8-159
with /SEQUENCE qualifier (L), 7-81 to 7-83, 8-175
with WRITE /SEQUENCE (L), 7-82, 8-292
key-name (L)
with DEFINE KEY (L), 7-26, 8-65
with SHOW KEY (L), 6-14 to 6-16, 7-27, 8-221
macro-name (L)
with DEFINE MACRO (L), 7-43 to 7-48, 8-68
n (L)
with /BRIEF qualifier (L), 4-46, 8-15
with /DUPLICATE qualifier (L), 4-46, 8-88
with SET WRAP (L), 8-209
with TAB ADJUST (L), 7-74 to 7-75, 8-265
;nokeypad-command(s) (L)
with CHANGE (L), 5-13
number (L)
with SET LINES (L), 8-186
with SET TAB (L), 8-202
number (N), 5-3
with ASC (ASCII) (N), 5-41 to 5-42, 8-9
prompt-type (L)
with SET PROMPT (L), 8-192
with SHOW PROMPT (L), 8-228
range (L), 4-2, 4-4 to 4-10, 4-14, 8-152
BEFORE, 4-5, 4-8
BEGIN, 4-5, 4-7
END, 4-5, 4-7
LAST, 4-5, 4-8
line number, 4-5, 4-6
percent sign (%) with, 4-9, 4-16
period (.), 4-5, 4-6
with <null> (L), 4-30
with TYPE (L), 4-29
REST, 4-5, 4-8
SELECT, 4-5, 4-9, 4-40, 5-30
string, 4-5 to 4-7
minus sign (–) with, 4-6
symbol
ALL, 4-5, 4-6
AND, 4-5
colon (:), 4-5
comma (,), 4-5
FOR n, 4-5
minus sign (–), 4-5
number (n), 4-5
number sign digit (# n), 4-5
plus sign (+), 4-5
THRU, 4-5

Specifier, range (L), (cont.)
WHOLE, 4-5, 4-8
with SUBSTITUTE (L), 4-20
with CHANGE (L), 5-13, 8-18
with DELETE (L), 4-36, 8-75
with FILL (L), 4-43 to 4-44, 8-102
with FIND (L), 4-16 to 4-18, 4-30, 8-108
with INCLUDE (L), 4-39, 8-120
with INSERT (L), 4-37, 8-121
with <null> (L), 4-9, 4-16 to 4-18, 4-30, 8-132
with PRINT (L), 4-41, 8-146
with /QUERY qualifier (L), 4-47
with REPLACE (L), 4-35 to 4-36, 4-38, 8-157
with RESEQUENCE (L), 4-41 to 4-43, 8-159
with SUBSTITUTE (L), 4-19, 4-19 to 4-23, 8-255
with SUBSTITUTE NEXT (L), 4-19, 4-21 to 4-23
with TAB ADJUST (L), 7-74 to 7-75, 8-265
with TYPE (L), 4-16 to 4-18, 4-30, 8-277
with WRITE (L), 4-40, 8-292
range-1 (L), 4-4
with COPY (L), 4-33 to 4-35, 8-33
with MOVE (L), 4-31 to 4-33, 8-128
range-2 (L), 4-4
with COPY (L), 4-33 to 4-35, 8-33
with MOVE (L), 4-31 to 4-33, 8-128
repeat count (N), 5-4
sign (+ | −) (N), 5-2 to 5-4, 5-6
with APPEND (N), 5-35 to 5-36, 8-8
with CHGC (change case) (N), 5-27, 8-21
with CHGL (change case lower) (N), 5-27, 8-22
with CHGU (change case upper) (N), 5-27, 8-23
with CUT (N), 5-31, 8-61
with D (delete) (N), 5-20 to 5-22, 8-62
with FILL (N), 5-40 to 5-41, 8-104
with "move" (N), 5-18 to 5-20, 8-129
with R (replace) (N), 5-24, 8-151
with S (substitute) (N), 5-24 to 5-26, 8-165
with SN (substitute next) (N), 5-26 to 5-27, 8-245
with SSEL (search and select) (N), 5-29, 5-30, 8-248
with TADJ (tab adjust) (N), 8-267
string (K), 8-250
Specifier, (cont.)
string (L), 8-250
with DEFINE KEY (L), 7-27, 8-65
with quotation marks (',"), 6-1
with SET (L), 6-1
with SET ENTITY (L), 6-6, 8-181
with SET PROMPT (L), 8-192
with SET TEXT (L), 8-205
string (N), 5-3 to 5-5, 5-9, 8-250
with S (substitute) (N), 5-24 to 5-26
with SSEL (search and select) (N), 5-29, 5-30, 8-248
with XLATE (N), 8-293
string-1 (L)
with SUBSTITUTE (L), 4-19 to 4-23, 8-255
with SUBSTITUTE NEXT (L), 4-19, 4-21 to 4-23, 8-259
string-1 (N)
with S (substitute) (N), 5-24 to 5-26, 8-165
string-2 (L)
with SUBSTITUTE (L), 4-19 to 4-23, 8-255
with SUBSTITUTE NEXT (L), 4-19, 4-21 to 4-23, 8-259
string-2 (N)
with S (substitute) (N), 5-24 to 5-26, 8-165
subtopic (L)
with HELP (L), 4-11 to 4-12, 8-116
top (L)
with SET CURSOR (L), 8-180
topic (L)
with HELP (L), 4-11 to 4-12, 8-116
width (L)
with SET SCREEN (L), 8-196
Specifier (RSTS/E)
DCL EDIT, E-1, E-3 to E-6
CCL EDT, E-6 to E-9
Specifier (RSX-11M/M+)
DCL EDIT/EDT, F-1 to F-4
MCR EDT, F-4 to F-7
Specifier (VAX/VMS)
DCL EDIT, D-1 to D-5
SPECINS (K), 2-10, 3-53 to 3-55, 7-2, 7-21, 8-246
with SET ENTITY (L), 6-6
Square brackets ([ ])
cursor indicator, hardcopy change mode, 7-76
enclosing command options, 4-3, 5-3
SR (select range) entity specifier (N), 5-6, 5-29 to 5-30

SSEL (search and select) (N), 5-11, 5-29, 5-30, 8-248
search string with, 5-30
with sign (+ | –) specifier, 5-29, 5-30
with string specifier, 5-4, 5-29, 5-30
Starting an editing session, 2-14, 3-6 to 3-8, 4-13, 5-12 to 5-13
Startup command file, 2-7, 5-42, 7-1 to 7-11, D-3, D-7, E-3, E-6 to E-9, F-2, F-5 to F-7
comments in, 7-2
/STAY qualifier (L)
with TYPE (L), 4-16, 4-46, 4-49, 8-249, 8-277
Storage area, 2-8
See also buffer
See also file
STREAM with /FORMAT qualifier (RSTS/E), E-4
String, 2-3, 2-11, 3-32 to 3-36
delimiter
CTRL/@ in key definitions, 7-18
search, 2-10, 2-12, 3-49 to 3-52, 4-19, 4-22, 5-24, 6-9 to 6-12
case of letters, 2-12, 6-10 to 6-11
changing, 3-46 to 3-48, 5-27 to 5-28
CLSS (clear search string) (N), 5-40
diacritical marks in, 2-12, 6-10 to 6-11
search (N)
with SSEL (search and select) (N), 5-30
substitute, 2-12, 3-49 to 3-52, 4-19, 4-22, 5-24
String delimiter, 2-11, 4-2, 5-2, 5-4
quotation marks (' ,"), 4-6, 5-30
String entity specifier (N), 5-6, 5-9, 5-24, 5-26
with D (delete) (N), 5-20
String range specifier (L), 4-5, 4-6 to 4-7
minus sign (–) with, 4-6
String specifier (K), 8-250
String specifier (L), 8-250
with DEFINE KEY (L), 7-27, 8-65
with quotation marks (' ,"), 6-1
with SET (L), 6-1
with SET ENTITY (L), 6-6, 8-181
with SET PROMPT (L), 8-192
with SET TEXT (L), 8-205
String specifier (N), 5-3, 5-4 to 5-5, 5-9, 8-250
with SSEL (search and select) (N), 5-29, 5-30, 8-248
with XLATE (N), 8-293
String-1 specifier (L)
with SUBSTITUTE (L), 4-19 to 4-23, 8-255
with SUBSTITUTE NEXT (L), 4-19, 4-21 to 4-23, 8-259
String-1 specifier (N)
with S (substitute) (N), 5-24 to 5-26, 8-165
String-2 specifier (L)
with SUBSTITUTE (L), 4-19 to 4-23, 8-255
with SUBSTITUTE NEXT (L), 4-19, 4-21 to 4-23, 8-259
String-2 specifier (N)
with S (substitute) (N), 5-24 to 5-26, 8-165
SUBS (K), 3-49 to 3-52, 7-21, 8-252
with DELETE buffer, 6-13
SUBSTITUTE (L), 4-1, 4-18 to 4-23, 8-255
delimiter, 4-19, 4-47
slash (/), 4-19
with /BRIEF qualifier, 4-19, 4-46 to 4-47
with /NOTYPE qualifier, 4-19, 4-46, 4-47
with /QUERY qualifier, 4-19, 4-46 to 4-48
with string-1 specifier, 4-19 to 4-23
with string-2 specifier, 4-19 to 4-23
with WHOLE range specifier, 4-20
Substitute buffer, 2-8, 4-19, 5-26, 6-12
SUBSTITUTE NEXT (L), 4-1, 4-18 to 4-19, 4-21 to 4-23, 8-259
with string-1 specifier, 4-19, 4-21 to 4-23
with string-2 specifier, 4-19, 4-21 to 4-23
Substitute string, 2-12, 3-49 to 3-52, 4-19, 4-22, 5-24, 5-26 to 5-27
Substituting, 2-12, 3-49 to 3-52, 4-18 to 4-23, 5-24 to 5-27, 6-9
delimiter character, 2-12, 4-19
keypad mode, 3-49 to 3-52
Subtopic specifier (L)
with HELP (L), 4-11 to 4-12, 8-116
Symbols for characters, 3-54 to 3-55, G-1 to G-6
Syntax, 4-2, 5-2
System
☞symbol for, 2-5
command level, 2-2, 5-13
operating, 2-2, 2-16
prompt, 2-2, 2-5
System interruption, 2-17, 2-18

T

Tab, horizontal
with V (vertical) entity specifier (N), 5-10
See also CTRL/I
TAB (K), 2-14, 3-52 to 3-53, 7-14 to 7-15, 7-23, 7-25, 7-28, 7-62 to 7-65, 7-68 to 7-73, 8-261
TAB (L), 2-14, 4-44 to 4-45, 7-62
TAB (N), 2-14, 5-11, 7-62, 8-263

TAB ADJUST (L), 4-2, 7-73 to 7-75, 8-265
with buffer specifier, 7-74 to 7-75
with minus sign (–), 7-74 to 7-75
with n (tab level count) specifier, 7-74 to 7-75
with range specifier, 7-74 to 7-75
Tab adjust function, 7-67 to 7-68
Tab character (horizontal) – CTRL/I, 4-44, 7-63
Tab compute function, 7-67
Tab decrement function, 7-67
Tab increment function, 7-66 to 7-67
TAB key, 3-2, 3-3, 3-52 to 3-53, 4-44, 6-14
See TAB (K), TAB (L)
Tab key (LK201 keyboard), 8-261
See also TAB (K)
Tab level count, 7-66 to 7-75
Tab stop, 3-52 to 3-53, 4-44, 7-63
initial, 7-63
preset, 7-62
Tabbing, 7-62 to 7-75
keypad, 3-52 to 3-53
TADJ (tab adjust) (N), 5-11, 7-62, 8-267
TC (tab compute) (N), 5-11, 7-62, 8-268
TD (tab decrement) (N), 5-11, 7-62, 8-270
Temporary file – .TMP, D-8, F-9
Tentative file, E-10
Terminal, 2-2, 2-4, C-1 to C-3, D-6, F-8
See also VT100, VT52, Hardcopy, LK201 keyboard, SET TERMINAL (L)
asynchronous, C-1
autorepeat suppression, D-6, E-9, F-9
eightbit graphics, D-6, E-9, F-9
escape sequences, D-6, E-9, F-9
GIGI (VK100), C-1
hardcopy, 2-14, 4-1, C-2
internal editing features, D-6, E-9, F-9
LK201 keyboard, 3-4, C-1
screen, 2-14, 4-1, C-1
scrolling regions, D-6, E-9, F-9
VT52, 3-1, 3-3, 5-1, C-1
keyboard, 3-4
keypad, 2-14, 3-5
keypad HELP screen display, 3-11
VT100, 3-1 to 3-3, 5-1, C-1
keyboard, 3-4
keypad, 2-14, 3-5
keypad HELP screen display, 3-11
Terminal bell, 5-40
Text entity
See entity
Text file, 2-2, 2-6
See also file
TGSEL (toggle select) (N), 5-11, 5-29, 5-30, 8-272
THRU range specifier symbol (L), 4-5
TI (tab increment) (N), 5-11, 7-62, 8-273
.TMP – temporary file, D-8, F-9
TOP (K), 3-14, 3-16, 7-21, 8-275
TOP (N), 5-11, 5-39, 8-276
with "move" (N), 5-39
Top of buffer, 3-16
Top specifier (L)
with SET CURSOR (L), 8-180
Topic specifier (L)
with HELP (L), 4-11 to 4-12, 8-116
Transmission rate – data, C-2
TYPE (L), 4-1, 4-16 to 4-18, 8-277
with /BRIEF qualifier, 4-16, 4-46 to 4-47
with buffer specifier, 4-30
with period (.) range specifier, 4-29
with range specifier, 4-30
with /STAY qualifier, 4-16, 4-46, 4-49

U

UND C (K), 3-18 to 3-20, 7-21, 8-279
UND L (K), 3-18, 3-24 to 3-29, 7-21, 8-281
UND W (K), 3-18, 3-20 to 3-24, 7-21, 8-286
UNDC (undelete character) (N), 5-11, 5-22 to 5-24, 8-280
with count specifier, 5-22 to 5-24
Undelete functions (K), 3-18 to 3-30
Undeleting characters, 3-18 to 3-20, 5-22 to 5-24
Undeleting lines, 3-24 to 3-29, 5-22 to 5-24
Undeleting text
keypad mode, 3-18 to 3-30
nokeypad mode, 5-22 to 5-24
Undeleting words, 3-20 to 3-24, 5-22 to 5-24
Underscore (_), 4-19
in buffer names, 4-4
UNDL (undelete line) (N), 5-11, 5-22 to 5-24, 8-284
with count specifier, 5-22 to 5-24
UNDW (undelete word) (N), 5-11, 5-22 to 5-24, 8-288
with count specifier, 5-22 to 5-24
Up Arrow (K), 3-14, 3-15, 7-21, 8-290
Up Arrow (N), 8-290

V

V (vertical) entity specifier (N), 5-6, 5-9 to 5-10
with horizontal tab, 5-10
VARIABLE
with /FORMAT qualifier (RSTS/E), E-4

VAX/VMS, 2-15, D-1 to D-17
Version number, file, 2-7, D-1, F-8
Vertical (V) entity specifier (N), 5-6, 5-9 to 5-10
  with horizontal tab, 5-10
Vertical tab
  word boundary, 3-20, 6-7
VFC
  page, 7-81
  record format, 7-82, 7-83
  sequenced file, 7-81 to 7-82
VK100 (GIGI) terminal, C-1
VT52 terminal, 2-2, 2-4, 3-1, 3-3, 5-1, C-1, D-6, E-9, F-9
  HELP file, 7-85
  keyboard, 3-4
  keypad, 2-14, 3-5
  keypad HELP screen display, 3-11
  keypad HELP text, 7-87
  keypad key numbers, 6-14
VT100 terminal, 2-2, 2-4, 3-1 to 3-3, 5-1, C-1, D-6, E-9, F-9
  CHAR (K), 3-14
  HELP file, 7-85
  keyboard, 3-4
  keypad, 2-14, 3-5
  keypad HELP screen display, 3-11
  keypad HELP text, 7-87
  keypad key numbers, 6-14

W

W (word) entity specifier (N), 5-5
WHOLE range specifier (L), 4-5, 4-8
  with SUBSTITUTE (L), 4-20
WIDTH (VAX/VMS terminal characteristic), D-6
Width specifier (L)
  with SET SCREEN (L), 8-196
Width, screen, C-1, D-7, E-9, F-9
  VK100 terminal, C-1
Wildcard character (*) with HELP (L), 4-12
WORD (K), 3-14, 7-21, 8-291
Word (W) entity specifier (N), 5-5
Word boundary, 3-20, 6-7 to 6-8
Word entity, 2-9, 3-20, 6-7 to 6-8
Work file, 7-81
WRITE (L), 2-7, 2-8, 4-2, 4-38 to 4-40, 8-292, D-7, D-8, E-2 to E-3, F-7
  tentative file, E-10
  .TMP (temporary) file, D-8, F-9
  with buffer specifier, 4-4, 4-28, 4-40
  with file specification specifier, 4-40
  with range specifier, 4-40
  with /SEQUENCE qualifier (L), 2-13, 4-40, 4-46, 7-82 to 7-83

X

XLATE (N), 5-11, 8-293, D-15 to D-16
  with string specifier, 5-4
XON switch (RSTS/E), E-9

Y

Y (yes), /QUERY qualifier response (L), 4-47 to 4-48
Yes –Y, /QUERY qualifier response (L), 4-47 to 4-48

Z

^Z (CTRL/Z) (L), 4-13
^Z (CTRL/Z) (N), 5-18

EDT Editor Manual
AA-M476A-TK

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate which system you are using:

☐ VAX/VMS ☐ RSTS/E ☐ RSX-11M ☐ RSX-11M-PLUS Other _____

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
or Country

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

BSSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03061

Do Not Tear - Fold Here

Cut Along Dotted Line

digital

DIGITAL EQUIPMENT CORPORATION